Lab 10 Guide:

First, create the troubleshoot.py file that will run all the troubleshooting code. Code will be explained in the video, but it is located in Lab4/troubleshoot.py

import os

import re

import pyshark

from csv import DictWriter

import subprocess

import time

import csv

from netmiko import ConnectHandler

def getSyslog():

    file = "traps.pcap"

    cap = pyshark.FileCapture(file)

    traps = []

    i = 1

    field\_names = ['ID', 'Router', 'Message', 'Level']

    with open('Syslog.csv', 'w') as f\_object:

        for packet in cap:

            dictwriter\_object = DictWriter(f\_object, field\_names)

            if 'Syslog' in packet and packet.Syslog.level <= "5":

                traps.append(str(packet.Syslog))

                #match = re.search(r'Syslog message id:\s\*(.\*)', str(packet.Syslog))

                match = re.search(r'%(.\*)', str(packet.Syslog))

                if not match:

                    continue

                result = match.group(1)

                syslog\_dict = {'ID':i, 'Router': packet.Syslog.hostname, 'Message':result, 'Level': packet.Syslog.level}

                dictwriter\_object.writerow(syslog\_dict)

            i+=1

#getSyslog()

def extract\_interface\_states(csv\_file):

    extracted\_data = []  # List to store extracted interface and state

    interface\_pattern = re.compile(r'Interface Ethernet(\S+),', re.IGNORECASE)

    state\_pattern = re.compile(r'state\s+to\s+(up|down)', re.IGNORECASE)

    with open(csv\_file, 'r') as file:

        reader = csv.reader(file)

        for row in reader:

            if len(row) < 3:  # Skip rows without enough data

                continue

            # Extract the message field from the row

            message = row[2]

            # Search for the interface and state

            interface\_match = interface\_pattern.search(message)

            state\_match = state\_pattern.search(message)

            if interface\_match:

                interface = interface\_match.group(1)

                state = state\_match.group(1) if state\_match else "unknown"

                extracted\_data.append({"Interface": interface, "State": state, "Hostname": row[1]})

    return extracted\_data

def sshInfo():

    csv\_file = "/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab4/devices.csv"

    data = {}

    with open(csv\_file, "r") as file:

        reader = csv.DictReader(file)

        for row in reader:

            router\_name = row["hostname"]

            router\_data = {

                "device\_type": row["device\_type"],

                "ip": row["ip"],

                "username": row["username"],

                "password": row["password"]

            }

            data[router\_name] = router\_data

    return data

def config\_interface(ip, user, password, interface):

    arista\_device = {

        'device\_type': "arista\_eos",

        'host': ip,  # Replace with the device IP address

        'username': user,    # Replace with your username

        'password': password, # Replace with your password

    }

    try:

        # Establish the connection

        connection = ConnectHandler(\*\*arista\_device)

        print(f"Connected to {arista\_device['host']}")

        # Enter enable mode (if required)

        if connection.check\_enable\_mode() is False:

            connection.enable()

        # Send the command to configure the interface

        commands = [

            'interface ethernet'+interface,  # Enter interface configuration mode

            'no shutdown'           # Bring up the interface

        ]

        output = connection.send\_config\_set(commands)

        # Print the command output

        print("Configuration output:")

        print(output)

        # Close the connection

        connection.disconnect()

        print("Connection closed.")

    except Exception as e:

        print(f"An error occurred: {e}")

def interface\_no\_shut(interface, hostname):

    routers = sshInfo()

    extract\_interface\_states("Syslog.csv")

    for i in routers:

        if hostname == i:

            config\_interface(routers[i]['ip'], routers[i]['username'], routers[i]['password'], interface)

def find\_dst\_ip(hostname, interface):

    csv\_file = "/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab4/devices.csv"

    routers = sshInfo()

    for i in routers:

        if hostname == i:

            ip = routers[i]['ip']

            username = routers[i]['username']

            password = routers[i]['password']

    device = {

        'device\_type': "arista\_eos",

        'host': ip,

        'username': username,

        'password': password

    }

    command = "show ip int br"

    with ConnectHandler(\*\*device) as net\_connect:

        net\_connect.enable()

        output = net\_connect.send\_command(command)

        print(output)

    match = re.search(r"Ethernet"+interface+"\s+(\S+)", output)

    if match:

        ip\_address\_with\_subnet = match.group(1)

        ip\_address = ip\_address\_with\_subnet.split('/')[0]  # Split and get only the IP address

        print(f"The IP address for Ethernet1.10 is: {ip\_address}")

        return ip\_address

    else:

        print("Ethernet1.10 not found")

def get\_ip\_connectivity(hostname, interface):

    csv\_file = "/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab4/devices.csv"

    dst\_ip = find\_dst\_ip(hostname, interface)

    devices = ["R1", "R2", "R3", "R4"]

    routers = sshInfo()

    ping\_success = {}

    for i in devices:

        if hostname != i:

            ip = routers[i]['ip']

            username = routers[i]['username']

            password = routers[i]['password']

            device = {

                'device\_type': "arista\_eos",

                'host': ip,

                'username': username,

                'password': password

            }

            command = "ping "+str(dst\_ip)

            with ConnectHandler(\*\*device) as net\_connect:

                net\_connect.enable()

                output = net\_connect.send\_command(command)

            match = re.search(r'(\d+) packets transmitted, (\d+) received', output)

            if match:

                packets\_transmitted = int(match.group(1))

                packets\_received = int(match.group(2))

                if packets\_transmitted == packets\_received:

                    ping\_success[i] = "True"

                else:

                    ping\_success[i] = "False"

            else:

                ping\_success[i] = "False"

    return ping\_success

def fix\_interface\_state():

    getSyslog()

    seen = set()

    data = extract\_interface\_states("Syslog.csv")

    logs = []

    logs.append("Checking for down interfaces\n")

    for entry in data:

        if entry['State'] == "down":

            if entry['Interface'] in seen:

                continue

            seen.add(entry['Interface'])

            logs.append("Found Interface: "+entry['Interface']+" to be down\n")

            logs.append("Now doing a no shutdown on Interface: "+entry['Interface']+"\n")

            interface\_no\_shut(entry['Interface'], entry['Hostname'])

    time.sleep(1)

    getSyslog()

    data\_after = extract\_interface\_states("Syslog.csv")

    seen\_after = set()

    logs.append("Now checking in Syslogs that interface is up\n")

    for entry in data\_after:

        if entry['State'] == "up":

            if entry['Interface'] in seen\_after:

                continue

            seen\_after.add(entry['Interface'])

            logs.append("Interface: "+entry['Interface']+ " is now up\n")

    logs.append("Now checking IP connectivity from other devices\n")

    seen\_ip = set()

    for entry in data\_after:

        if entry['State'] == "up":

            if entry['Interface'] in seen\_ip:

                continue

            seen\_ip.add(entry['Interface'])

            pings = get\_ip\_connectivity(entry['Hostname'], entry['Interface'])

            for i in pings:

                if pings[i] == "True":

                    logs.append(i+" ping PASSED for device: "+entry['Hostname']+" at interface: "+entry['Interface']+"\n")

                else:

                    logs.append(i+ " ping FAILED for device: "+entry['Hostname']+" at interface: "+entry['Interface']+"\n")

    return logs

Next, create the tab on the website for troubleshooting both in the website.py file and a new file that was created called templates/troubleshoot.html

Here is the troubleshoot.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Troubleshoot</title>

    <style>

        body {

            background-color: #121212;

            color: #ffffff;

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

        }

        h1, h2, h3 {

            color: #bb86fc;

        }

        button {

            background-color: #bb86fc;

            color: #121212;

            padding: 10px 15px;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            font-size: 16px;

            margin-top: 10px;

        }

        button:hover {

            background-color: #9a67d9;

        }

        pre {

            background-color: #1e1e1e;

            padding: 15px;

            border-radius: 4px;

            overflow-x: auto;

            white-space: pre-wrap;

            margin-top: 10px;

            color: #e0e0e0;

        }

        .header {

            text-align: center;

            padding: 20px;

            background-color: #1e1e1e; /\* Header background \*/

            border-bottom: 2px solid #333; /\* Divider line \*/

        }

    </style>

</head>

<body>

    <div class="header">

        <h1>Logan's Netman Tool</h1>

    </div>

    <h1>Troubleshoot</h1>

    <form method="POST">

        <button type="submit">Troubleshoot</button>

    </form>

    <!-- Output Section -->

    {% if output %}

    <h3 class="my-4">Troubleshooting Results:</h3>

    <pre>{{ output }}</pre>

    {% endif %}

</body>

</html>

After these are created, add the tests for the Code coverage in the unit\_tests.py file:

class routerTests(unittest.TestCase):

    def test\_network\_ping\_test(self):

        self.assertTrue(network\_ping\_test())

    def test\_netconf(self):

        self.assertTrue(netconf\_test())

    def test\_snmp\_cpu(self):

        self.assertTrue(cpu\_test())

    def test\_snmp\_traps(self):

        self.assertTrue(traps\_test())

    def test\_snmp\_syslog(self):

        self.assertTrue(syslog\_test())

    def test\_create\_user\_pass(self):

        self.assertTrue(create\_user\_pass\_test())

    def test\_update\_router\_credentials(self):

        self.assertTrue(update\_router\_credentials\_test())

    def test\_configure\_arista\_device(self):

        self.assertTrue(configure\_arista\_device\_test())

    def test\_change\_passwords(self):

        self.assertTrue(change\_passwords\_test())

    def test\_createAccess(self):

        self.assertTrue(createAccess\_test())

    def test\_createCore(self):

        self.assertTrue(createCore\_test())

    def test\_createEdge(self):

        self.assertTrue(createEdge\_test())

    def test\_get\_neighborships(self):

        self.assertTrue(get\_neighborships\_test())

    def test\_get\_route\_table(self):

        self.assertTrue(get\_route\_table\_test())

    def test\_get\_cpu(self):

        self.assertTrue(get\_cpu\_test())

    def test\_get\_ip\_connectivity(self):

        self.assertTrue(get\_ip\_connectivity\_test())

    def test\_sshInfo(self):

        self.assertTrue(sshInfo\_test())

    def test\_pcap(self):

        self.assertTrue(pcap\_test())

    def test\_devices(self):

        self.assertTrue(devices\_test())

if \_\_name\_\_ == '\_\_main\_\_':

    data = [

            {"name": "SNMP.py", "count": SNMP\_count, "total": count\_functions\_in\_file("/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab2/SNMP.py")},

            {"name": "NETCONF.py", "count": NETCONF\_count, "total": count\_functions\_in\_file("/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab2/NETCONF.py")},

            {"name": "passwords.py", "count": passwords\_count, "total": count\_functions\_in\_file("/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab4/passwords.py")},

            {"name": "playbookCreation.py", "count": playbookCreation\_count, "total": count\_functions\_in\_file("/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab4/playbookCreation.py")},

            {"name": "troubleshooting.py", "count": troubleshooting\_count, "total": count\_functions\_in\_file("/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab4/troubleshooting.py")}

        ]

    with open("/home/student/Documents/CSCI5840\_Advanced\_Network\_Automation/Lab7/counts.csv", mode="w", newline="") as file:

        writer = csv.DictWriter(file, fieldnames=["name", "count", "total"])

Once these are done, simply run a tshark instance to capture any Syslog messages, you can leave this on for as long as you want or run it in the background:

tshark -w traps.pcap -i CR\_e1-1

After that, all you must do is press the troubleshoot button, and it will start to automatically troubleshoot problems in the network.

A black and white screen

Description automatically generated