Ali Abdelhamid
Ihab Agha
30/12/2022
Networking

Network Monitoring system

-Table of contents

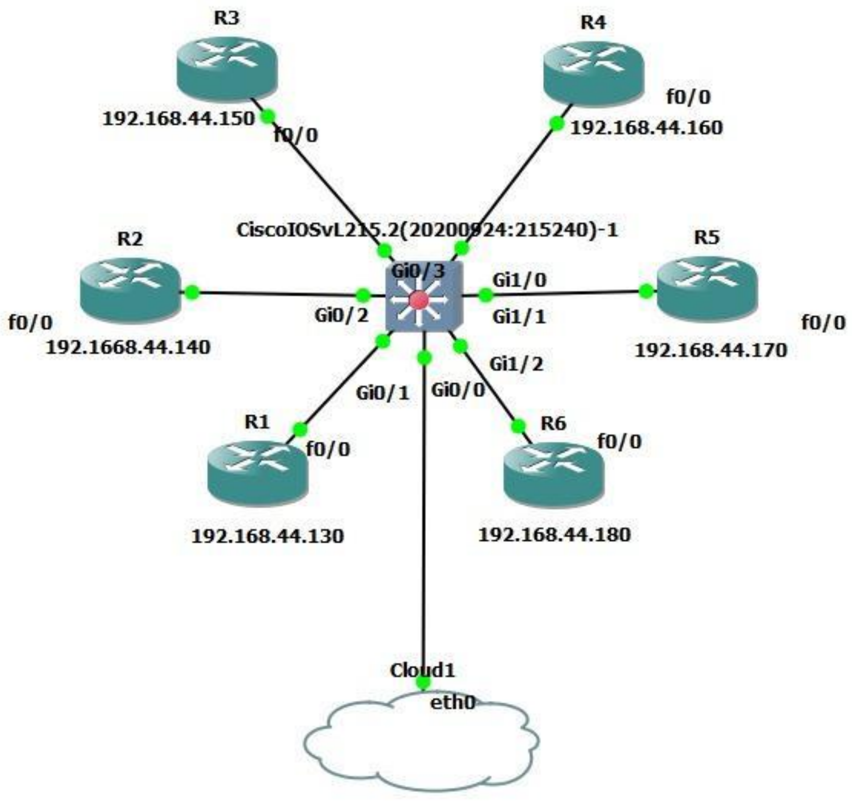
Introduction

This report will contain the exact steps I took in order to create this networking monitoring solution system for a company called FekkiSec, as they have just opened a new building and installed a number of routers on it, I was asked to install a security system onto the network to monitor all connected devices as well as select a designated router daily and block unknown mac addresses that may connect to the network, then start a network interface scanner on the router with the threat then output the results of the scanner to file on my computer. I mentioned all my steps, screenshots, code I developed, and screenshots of the output of the code

Objectives

The main objective of this monitoring tool is to increase the security on the system, by constantly changing the designated router daily, as well as every time the code restarts. It will also monitor the mac address table every hour and checks if any unknown device has connected then blocks it instantly. This was done mainly to prevent anyone who got initial access from ever accessing again.

Network Layout



This is a general overview of the organization's network infrastructure. The central hub of the system is in the server room and consists of a main switch. Six routers, located on different floors of the building, provide extensive internet coverage. The switch is then linked to the Cloud1 interface, which represents the ISP in this scenario. The connection to the cloud is established through a virtual machine running Gns3 on a local device. This configuration allows for the simulation of connectivity as if the device were on the same network as the organizations. Each router has its own standalone IP address, though all are on the same subnet as the virtual machine (Cloud1). The star topology was chosen for its suitability for the organization's needs and compatibility with the OSPF protocol.

Devices

Name	Model	IP	Usage
R1	Cisco c7200	192.168.44.130	Router on the network
R2	Cisco c7200	192.168.44.140	Router on the network
R3	Cisco c7200	192.168.44.150	Router on the network
R4	Cisco c7200	192.168.44.160	Router on the network
R5	Cisco c7200	192.168.44.170	Router on the network
R6	Cisco c7200	192.168.44.180	Router on the network
SW1	Cisco IOSvL215.2	192.168.44.120	The Main Switch
Cloud 1	GNS3 VM	192.168.44.128	Connection to local machine

All the routers mentioned on this table have undergone OSPF configuration and have SSH enabled on them as well. The switch has port security configured in addition to also having SSH enabled.

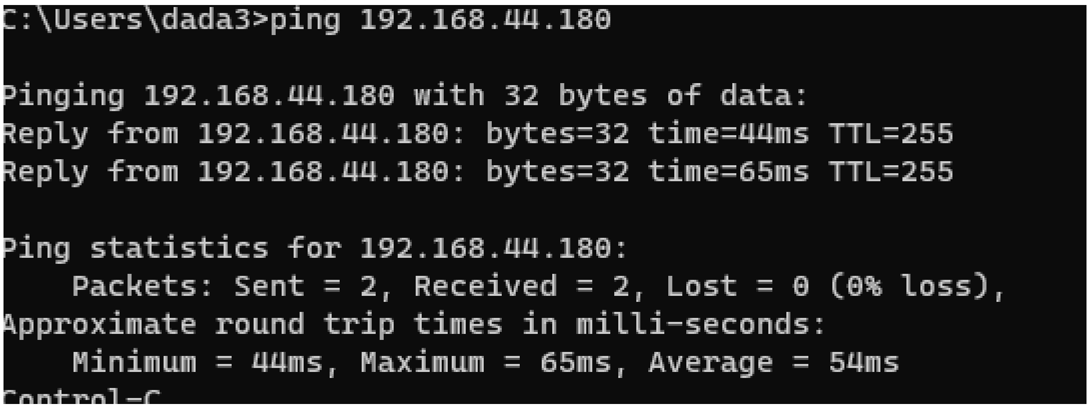
Configuration

In this explanation, I will detail the configuration commands I used to properly set up all the routers and switches, as well as how I was able to ping the internal routers from my local machine on my local network.
To begin, I applied similar commands to all the routers since they were largely identical with a few minor variations.

First, I accessed the console on each router and entered the configuration terminal using the command 'conf t'. I then specified the hostname for the router, such as 'R1' on the first router, 'R2' on the second router, and so on. I also executed the command 'no ip domain-lookup' on all the routers to disable continuous domain lookup when entering incorrect commands.
Next, I accessed the fastEthernet0/0 interface, the default interface used to connect the router to the switch and assigned an IP address to the router based on the subnet of the ISP. In this case, the IP of the ISP was 192.168.44.128, so the router's IP had to be in the range 192.168.44.xxx with a subnet mask of 255.255.255.0. For example, I set the IP to 192.168.44.130 on 'R1', 192.168.44.140 on 'R2', and 192.168.44.150 on 'R3', and so on.

I then moved on to configuring OSPF on each router. Although there were slight differences between each router, the OSPF commands were generally the same. I used the command 'router ospf 1' to enable OSPF on the router and set the process ID to 1, followed by 'router-id 10.x.x.x' to specify the router ID. The x's were replaced with the appropriate numbers for each router (e.g., 10.1.1.1 for 'R1', 10.2.2.2 for 'R2', and so on). I also used the command 'network 192.168.44.0 0.0.0.255 area 0' on all routers to specify a range of IP addresses from 192.168.44.0 to 192.168.44.255, which represents a subnet belonging to OSPF area 0.

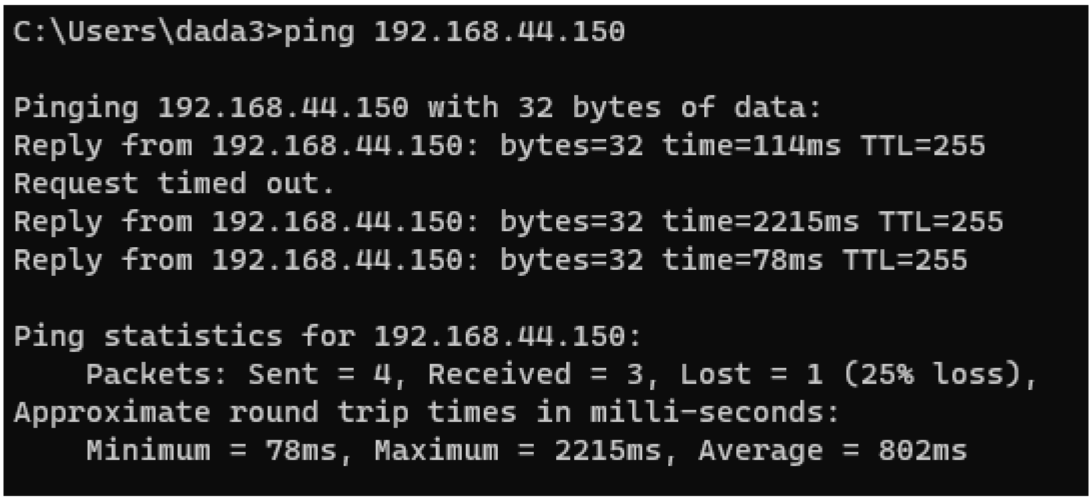
I then configured port security on the central switch. I set the configuration mode on VLAN 80, naming it 'VLAN NOTUSED'. I then went to all the unused interfaces (i.e., those not connected to a router) and applied the following commands: 'switchport mode access' to set the port to access mode, which is typically used to connect a single device, and 'switchport access vlan 80' to place the port in the 'VLAN NOTUSED'. I then shut down the interface using the command 'shutdown' and repeated these steps on all unused interfaces.
For the used ports (i.e., those connected to a router), I also set them to access mode using the command 'switchport mode access' and enabled port security on all of them using the command 'switchport port-security'. This also made the port only allow whitelisted MAC addresses onto the network and set the MAC address acquisition to be 'sticky', meaning that the port will retain blocked devices even after disconnection and reconnection. The remaining security measures for the switch were implemented in the code rather than manually.
After successfully assigning an IP to each router, I confirmed that my local network could see them, to do that I did a simple ping command on the command prompt on every router's IP. Here are some examples:



```
C:\Users\dada3>ping 192.168.44.180

Pinging 192.168.44.180 with 32 bytes of data:
Reply from 192.168.44.180: bytes=32 time=44ms TTL=255
Reply from 192.168.44.180: bytes=32 time=65ms TTL=255

Ping statistics for 192.168.44.180:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 44ms, Maximum = 65ms, Average = 54ms
Control-C
```



```
C:\Users\dada3>ping 192.168.44.150

Pinging 192.168.44.150 with 32 bytes of data:
Reply from 192.168.44.150: bytes=32 time=114ms TTL=255
Request timed out.
Reply from 192.168.44.150: bytes=32 time=2215ms TTL=255
Reply from 192.168.44.150: bytes=32 time=78ms TTL=255

Ping statistics for 192.168.44.150:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 78ms, Maximum = 2215ms, Average = 802ms
```

Security

I implemented an efficient security monitoring mac address, but what is the mac address?

The MAC (Media Access Control) address is a unique identifier that is assigned to each device on a network. It is used to identify the devices and facilitate communication between them. MAC addresses are typically represented in hexadecimal notation and are organized into a 48-bit value, with the first 24 bits representing the organizationally unique identifier (OUI) and the last 24 bits representing the device's identifier within the OUI.

In the script, the MAC addresses of the devices connected to the switches are checked to see if any unknown or unauthorized MAC addresses are present. To do this, the script connects to the switch and retrieves a list of all the MAC addresses on the switch using the "show mac address-table" command. This command displays the MAC address table for the switch, which includes the MAC addresses of all the devices that are currently connected to the switch, as well as the port on which each device is connected.

The script then parses the output of this command and looks for any MAC addresses that are marked as "Dynamic". These are typically MAC addresses that have been learned by the switch and are not listed in the switch's address table. When a switch receives a packet from a device that is not listed in its address table, it will typically add the MAC address of the device to the table as a dynamic entry.
If an unknown MAC address is detected on a switch, the script blocks the address using the "mac access-list extended" and "deny" commands. The "mac access-list extended" command is used to create an extended access list that can filter traffic based on the MAC addresses of the devices. The "deny" keyword is used to specify that the MAC address should be blocked. By using these commands, the script can block traffic from the unknown MAC address and prevent it from accessing the network. After blocking the unknown MAC address, the script captures packets on the router where the threat was detected using the "sniff" function from the "scapy" library in Python. Scapy is a powerful packet manipulation library for Python that allows you to create, send, and receive network packets. It is often used for network testing, debugging, and security-related tasks, such as packet capture and analysis. The "sniff" function is used to capture packets on the router where an unknown MAC address was detected. The function takes two arguments: "iface", which specifies the interface to use for packet capture, and "filter", which specifies a BPF (Berkeley Packet Filter) expression that determines which packets should be captured. In this case, the script specifies the "Ethernet" interface and a filter that specifies that only packets to or from the switch should be captured.

The captured packets are then saved to a file using the "wrpcap" function from the "scapy" library. This function takes two arguments: the name of the file to which the packets should be saved, and the packets to be saved. The packets are saved in the PCAP (Packet Capture) format, which is a common format for storing network packets. The use of MAC addresses and packet capture can be a useful tool for detecting and responding to security threats on a network. By checking for unknown MAC addresses and capturing packets on the affected router, it is possible to gather valuable information about the nature and source of the threat. This information can be used to identify and mitigate the threat, and to prevent similar threats from occurring in the future. As seen here the security implemented is quite efficient and will work to prevent any perpetrators from intervening with the daily workflow of the company

Budget

Component	Quantity	Price (estimate)	Total
Cisco 7200 router	6	1,000$	6,000$
Multi-access switch	1	1,000$	1,000$
Cloud	1	500$	500$
Total			7,500$

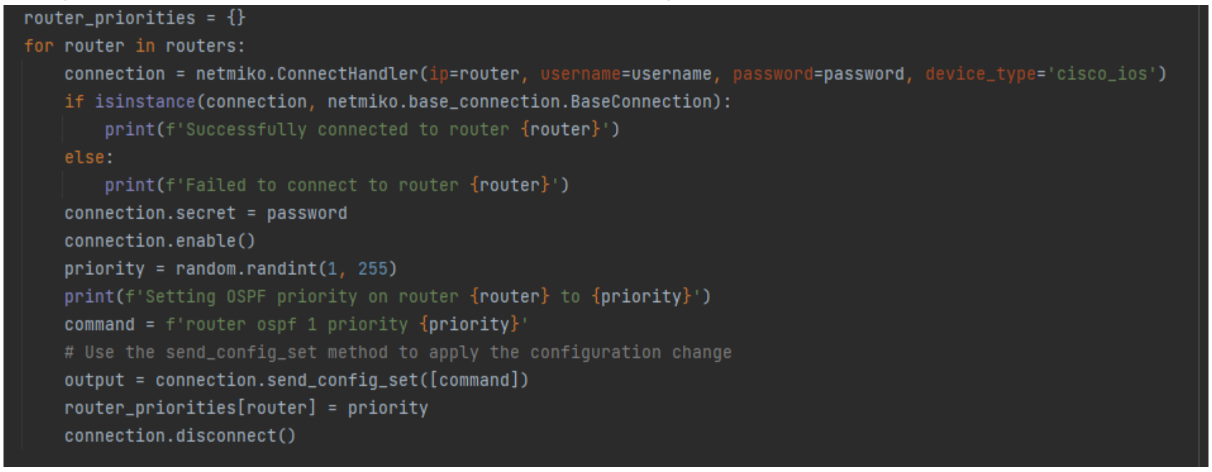
Technical Report

The script connects to Cisco routers and switches using the netmiko library to send commands and retrieve information over an SSH connection.

The script begins by defining two lists: routers and switches. The routers list contains the IP addresses of the routers in the network, and the switches list contains the IP addresses of the switches. The script also sets the username and password variables, which will be used to authenticate the connection to the devices.

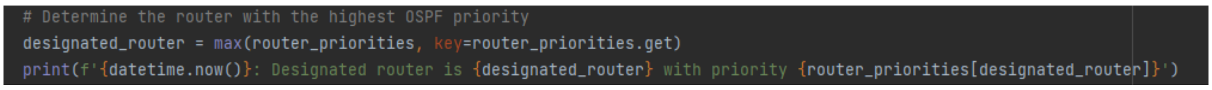
The script then enters an infinite loop that performs the following tasks:

1- Connects to each router in the routers list and sets the OSPF priority to a random value between 1 and 255 using the router ospf 1 priority command. This is done using the netmiko ConnectHandler and send_config_set methods.



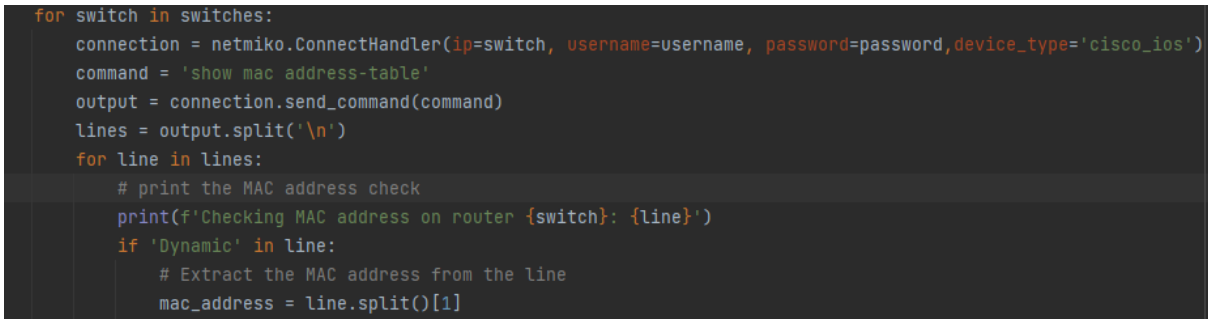
```python
router_priorities = {}
for router in routers:
    connection = netmiko.ConnectHandler(ip=router, username=username, password=password, device_type='cisco_ios')
    if isinstance(connection, netmiko.base_connection.BaseConnection):
        print(f'Successfully connected to router {router}')
    else:
        print(f'Failed to connect to router {router}')
    connection.secret = password
    connection.enable()
    priority = random.randint(1, 255)
    print(f'Setting OSPF priority on router {router} to {priority}')
    command = f'router ospf 1 priority {priority}'
    # Use the send_config_set method to apply the configuration change
    output = connection.send_config_set([command])
    router_priorities[router] = priority
    connection.disconnect()
```

2- Determines the router with the highest OSPF priority by iterating over the router_priorities dictionary and using the max function with the key parameter set to router_priorities.get. The designated router is then printed to the console.



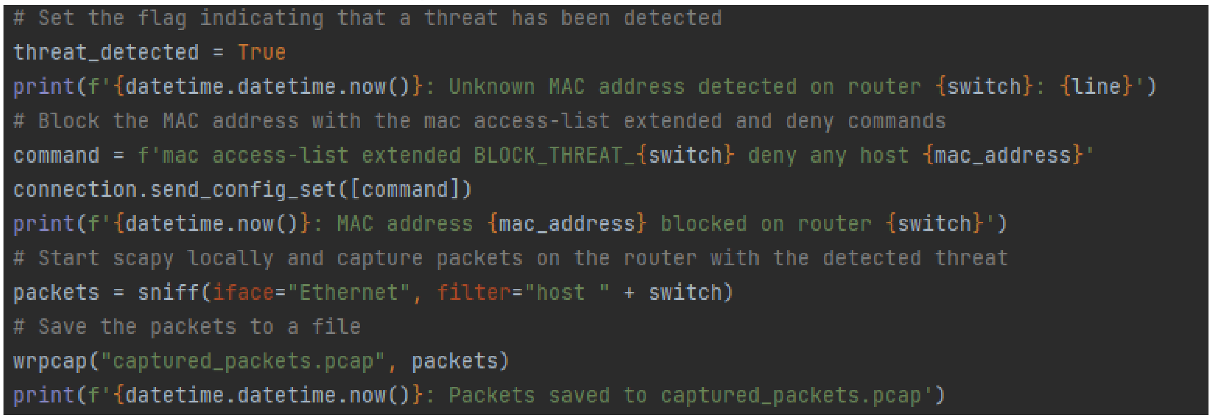
```python
# Determine the router with the highest OSPF priority
designated_router = max(router_priorities, key=router_priorities.get)
print(f'{datetime.now()}: Designated router is {designated_router} with priority {router_priorities[designated_router]}')
```

3- Connects to the switch and retrieves the MAC address table using the show mac address-table command. The output of this command is split into lines and iterated over. If a line contains the word "Dynamic", which indicates that the MAC address is unknown, the script sets a flag indicating that a threat has been detected.



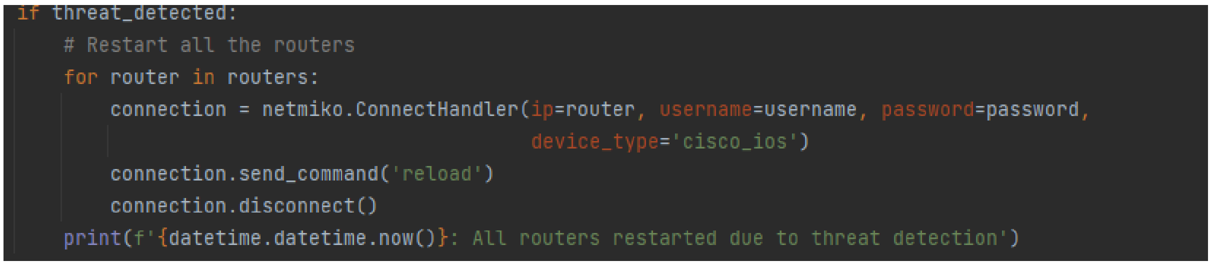
```
for switch in switches:
    connection = netmiko.ConnectHandler(ip=switch, username=username, password=password,device_type='cisco_ios')
    command = 'show mac address-table'
    output = connection.send_command(command)
    lines = output.split('\n')
    for line in lines:
        # print the MAC address check
        print(f'Checking MAC address on router {switch}: {line}')
        if 'Dynamic' in line:
            # Extract the MAC address from the line
            mac_address = line.split()[1]
```

4- If a threat was detected, the script blocks the unknown MAC address on the switch using the mac access-list extended command and starts a packet capture using scapy on the switch where the threat was detected. The captured packets are then saved to a file called "captured_packets.pcap".



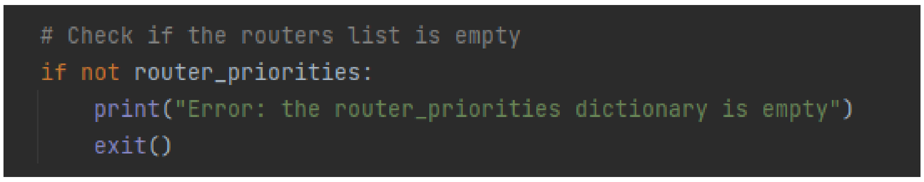
```
# Set the flag indicating that a threat has been detected
threat_detected = True
print(f'{datetime.datetime.now()}: Unknown MAC address detected on router {switch}: {line}')
# Block the MAC address with the mac access-list extended and deny commands
command = f'mac access-list extended BLOCK_THREAT_{switch} deny any host {mac_address}'
connection.send_config_set([command])
print(f'{datetime.now()}: MAC address {mac_address} blocked on router {switch}')
# Start scapy locally and capture packets on the router with the detected threat
packets = sniff(iface="Ethernet", filter="host " + switch)
# Save the packets to a file
wrpcap("captured_packets.pcap", packets)
print(f'{datetime.datetime.now()}: Packets saved to captured_packets.pcap')
```

5- As well as restarting all the routers in the routers list using the reload command.



```
if threat_detected:
    # Restart all the routers
    for router in routers:
        connection = netmiko.ConnectHandler(ip=router, username=username, password=password,
                                            device_type='cisco_ios')
        connection.send_command('reload')
        connection.disconnect()
    print(f'{datetime.datetime.now()}: All routers restarted due to threat detection')
```

The script also includes some basic error handling, such as checking if the connection to the device was successful and if the router_priorities dictionary is empty.



```
# Check if the routers list is empty
if not router_priorities:
    print("Error: the router_priorities dictionary is empty")
    exit()
```

References

1- datetime (n.d.). datetime - Date and time value manipulation and formatting. Retrieved from https://docs.python.org/3/library/datetime.html

2- netmiko (n.d.). Netmiko. Retrieved from https://pynet.twb-tech.com/lib/netmiko/index.html

3- scapy (n.d.). Scapy. Retrieved from https://scapy.net/

4- random (n.d.). random - Generate pseudo-random numbers. Retrieved from https://docs.python.org/3/library/random.html

5- Cisco Systems, Inc. (n.d.). OSPF Designated Router Priority. Retrieved from https://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fsospf.html

6- Cisco Systems, Inc. (n.d.). mac access-list. Retrieved from https://www.cisco.com/c/en/us/td/docs/ios/12_2/security/command/reference/fsecur_r/srfmacac.html

7- Cisco Systems, Inc. (n.d.). reload. Retrieved from https://www.cisco.com/c/en/us/td/docs/ios/12_2/configfun/command/reference/ffun_r/frf011.html

8- Cisco Systems, Inc. (n.d.). show mac address-table. Retrieved from https://www.cisco.com/c/en/us/td/docs/ios/12_2/switch/command/reference/fsw_r/srfwmac.html

9- OSPF (n.d.). Retrieved from https://en.wikipedia.org/wiki/Open_Shortest_Path_First