**Network Engineering Capstone  
Functionality Report**

# Introduction

*Provide a functionality report detailing the 10 test-case scenarios used to verify the utility of your network project. Seven of the test-case scenarios must be from the provided predefined list, with the remaining three test cases created by you. The functionality report should be written so that a networking peer could replicate the steps for a successful test of your networking solution.*

|  |  |
| --- | --- |
| **Student Name** | Emily Parker |
| **WGU Student ID** | 011200197 |
| **WGU Student Email** | epar255@wgu.edu |

# Test Case #1: Device Discovery and Reachability

*Your network solution must include multiple network segments with access controls that allow traffic from a device on one network to access the resources of a device on another network. Similarly, there must be devices on one network that cannot access resources on a different network.*

## Functionality

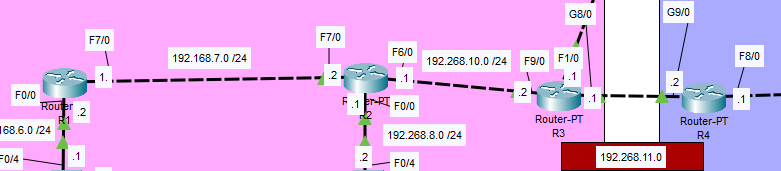
*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The functionality of this test case is exhibited by pinging from R1, at Fenrir site 1 to R4, at Fenrir site 2. For traffic to successfully flow from R1 to R4, there needs to be proper routes configured. Static routes were configured and were verified to be working correctly. Tools used include the “IP route” command, and the ping command which utilizes ICMP echo requests/replies. Allowing certain devices to access/not access resources will be configured in test case #2 (ACL’s) instead of here. Relevant devices include R1, R2, R3, and R4. Relevant subnets include 192.168.7.0 /24, 192.168.10.0 /24, and 192.168.11.0 /24.

Note: “*Similarly, there must be devices on one network that cannot access resources on a different network.”* This functionality is part of test case #2. As such, it will be implemented in said test case.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*



## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

The testing method used here was the ping command. By pinging R4 from R1, I was able to verify that traffic is able to be forwarded between site 1 and 2. Metrics of success include the success rate of the pings being 5/5, meaning there was no loss, and all packets reached the destination.

## Process List

*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*

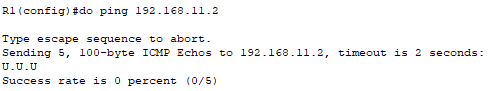
After static IP’s were assigned, the next step to test reachability was to have routes configured. Test case #8 will cover OSPF configuration for dynamic routing. For now static routes will do, they will be deleted afterwards.



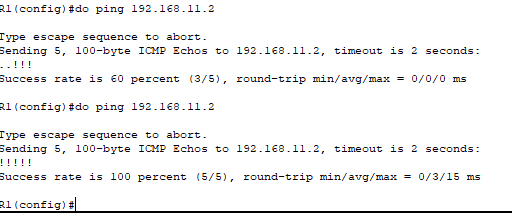


Ping from 192.168.7.0 to 192.168.11.0 should still fail, routes haven’t been configured on the other routers.

It does fail, this is expected behavior.



After configuring the other two static routes on R2 and R3, the ping between Fenrir 1 and Fenrir 2 was successful.



# Test Case #2: Administering an Access Control List for Guest Access

*Your network must utilize an access control list that allows guest access. Guest access should be limited to internet traffic only.*

## Functionality

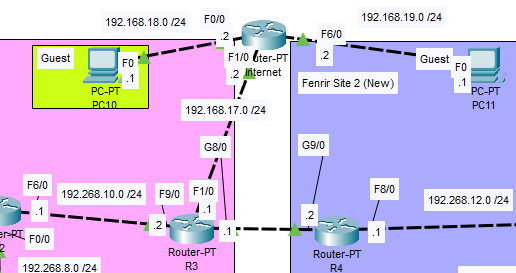
*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

Note: PC’S on the guest subnets have static IP’S only temporarily for this test case. They will acquire dynamic IP’S later in test case #4.

Functionality exhibited here includes the PC’s on each Guest subnet being able to access the Internet and only the Internet. The router labeled “Internet”; hostname “ISP-Router” is really a stand-in. Packet Tracer has limited functionality, so the PC’s cannot actually access the Internet. Access to internal network resources is denied by the use of a standard numbered ACL. Commands I have used here include the “ip access-list [*number*]” command to create the ACL, “access-list [*number*] permit/deny [*IP*] [*wildcard mask*]” to add an ACE to the list, and the “ip access-group [*number*] in/out” command to apply the ACL inbound or outbound on the interface. It is very important to use a wildcard mask instead of a standard subnet mask, or else it will not work correctly. For clarity, you must create the ACL in global config mode, and apply it in interface config mode.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*



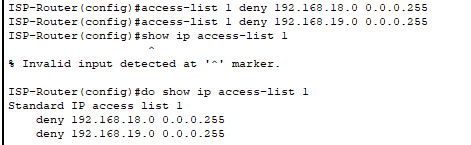
## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

 The testing method I used to verify functionality of this test case is to use the ping command to verify the PC’s on the guest subnet cannot access internal network resources. I applied the ACL outbound on ISP-Router’s F1/0 interface so that when PC10 and PC11 try to communicate to the internal network, they are simply blocked from doing so. On top of that, **ONLY** those two subnets are blocked when ISP-Router checks its outbound ACL for matches. Pings from the guest PC’s to the internal network were successfully blocked, as shown below in the Process List.

## Process List

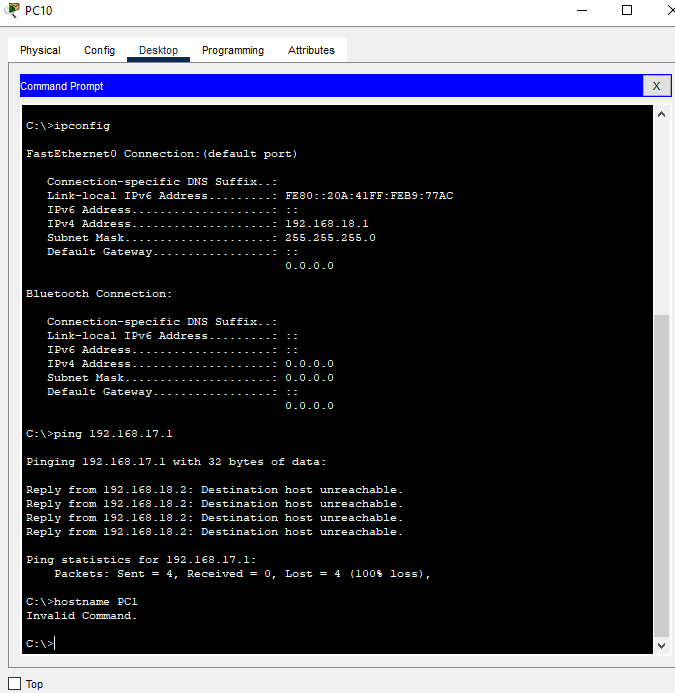
*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*



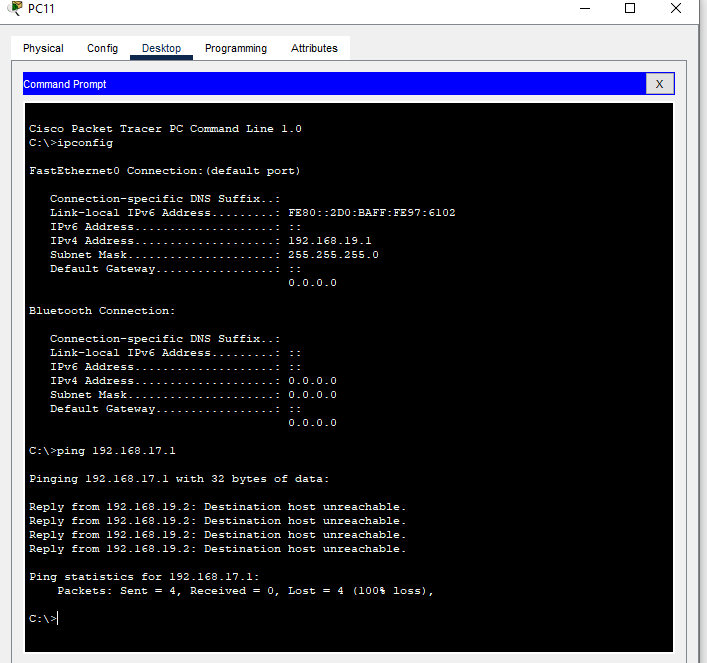
Commands to add ACE’s to the ACL, then a show command to list all current IP access-lists. NOTE: The command to create a standard numbered ACL is “ip access-list [*number*]”. These commands must be run in global config mode.



Switching to interface F1/0, and applying the ACL outbound. The command here is a bit different, as you specify “access-group” rather than “access-list” This is done in interface config mode.



Verifying that pings from PC10 to ISP-Router’s F1/0 interface fail. This behavior is expected, and wanted. This proves that the configuration and application of the ACL is correct.



Again verifying that pings were unsuccessful, this time from PC11 to ISP-Router’s F1/0 interface. This behavior is expected, and wanted.

# Test Case #3: Security Compliance—Log-in Banners

*Display a log-in banner when accessing each device on the network. The log-in banner should notify users of an acceptable use policy (AUP) or other security-based policies when attempting to log into the network.*

## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The functionality exhibited in this test case is meant to show the company Acceptable Use Policy to users when logging onto network devices. Relevant tools include the “banner motd” command, followed by a delimiter character of your choosing. This will allow you to write your login message, and then end it with the delimiter character. Relevant devices include every L2 switch, L3 switch, and router in the network. If Packet Tracer supported the functionality, the login banner would also be present on every end user device in the network.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*

A blue square with black lines and green triangles

Description automatically generated

## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

 The testing method in this case is very simple. Open a CLI connection on the selected device, and the AUP will show up.

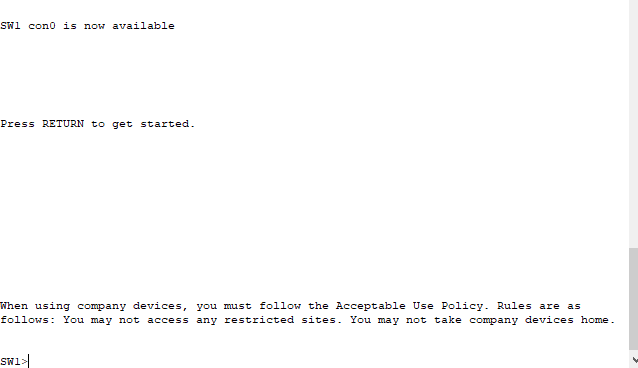
## Process List

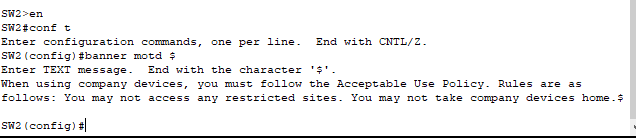
*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*

The first step in creating login banners for the network devices is to use the “banner motd” command, followed by a delimiter of your choosing. This will allow the device to know which part is the command and which part is the login banner. NOTE: “motd” stands for message of the day.Then you can write your banner, and add the delimiter again.

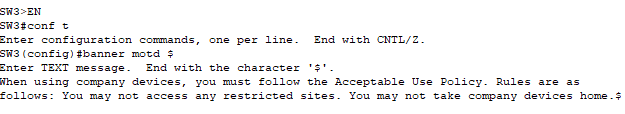


SW1 login banner configuration

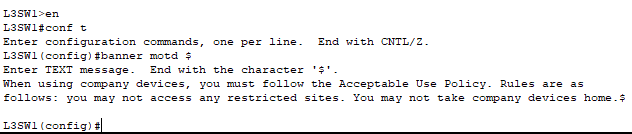




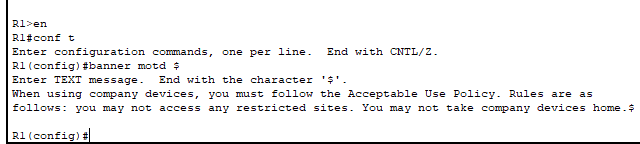
SW2 login banner configuration



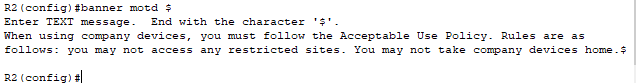
SW3 login banner configuration



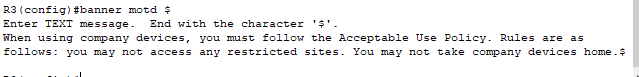
L3SW1 login banner configuration



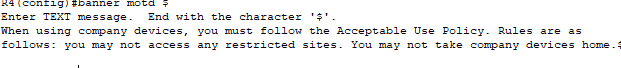
R1 login banner configuration



R2 login banner configuration



R3 login banner configuration



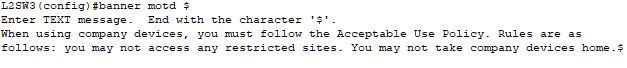
R4 login banner configuration



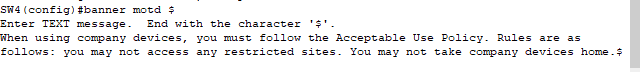
R5 login banner configuration



R6 login banner configuration



L3SW3 login banner configuration



SW4 login banner configuration



SW5 login banner configuration



SW6 login banner configuration

As seen here. when logging onto a device, the login banner that I have configured is displayed. Packet Tracer does not have the functionality required for configuring login banners on the PC’s.

# Test Case #4: Accessing External Resources—Routing and Traffic Security

*User devices on your network should have dynamic addresses that are assigned through DHCP unless they provide a service that requires a static address. You must also have at least one network resource that requires a static address.*

## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The desired functionality here is to have all end user devices to have IP addresses assigned dynamically by the use of DHCP. This way, devices that no longer need an IP address(ex. Device is powered off, or no longer in use) can release their IP’s back to the DHCP server for use by another device. This creates a smooth and dynamic IP allocation and release process. If necessary, devices can request a reservation so that they receive the same IP upon each boot up, for example. Network devices should receive static IP’s so we can reliably access and perform configuration changes with a previously known IP. For example, if each time you SSH into a device and the IP is different, it creates unneccsary confusion and effort. We want the process to be smooth and simple. Our guest networks should not communicate with the internal network, so they will receive IP’s from ISP-Router, configured as a DHCP server. The hosts on the internal network will receive IP’s from our internal DHCP servers. Relevant subnets include 192.168.3.0 /24, 192.168.4.0 /24, 192.168.5.0 /24, 192.168.18.0 /24, 192.168.19.0 /24 PC0, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, PC9, PC10, and PC11.

## Network Diagram or Segment

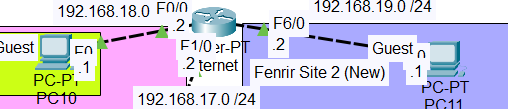
*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*

A computer icons with numbers and letters

Description automatically generated with medium confidence

A computer with a sign

Description automatically generated



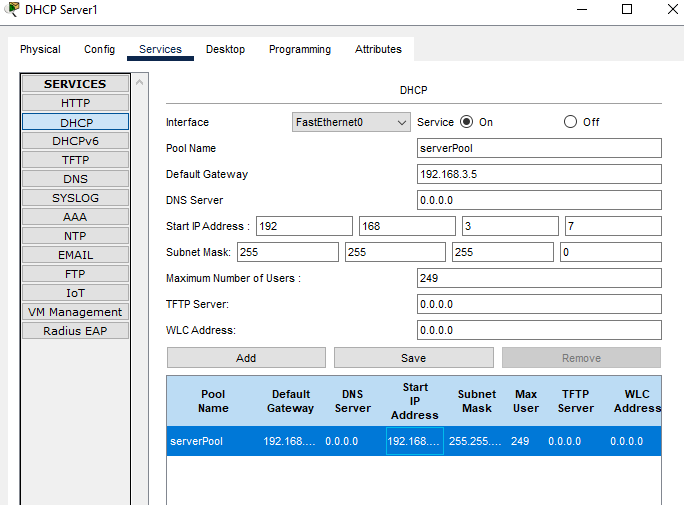
## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

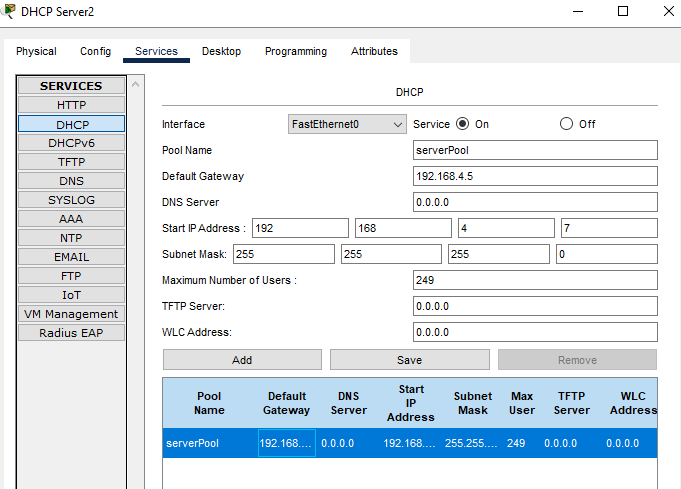
The testing method I used to verify functionality of DHCP was to use the commands “ipconfig”, “ipconfig /release”, and “ipconfig /renew.” The ipconfig command by itself will show information such as your current IP address, subnet mask, and default gateway. The release addition to the command allows the client to relase their IP back to the DHCP server. Finally, the renew part of the command allows the client to query the DHCP server for a new IP address. First I used ipconfig by itself to verify that the client does in fact not have a current IP address, or has an APIPA address, which would point to the DHCP server not working. I then added /release to be positive the client did not have an address. Then, I used the /renew command to allow the client to acquire an IP address.

## Process List

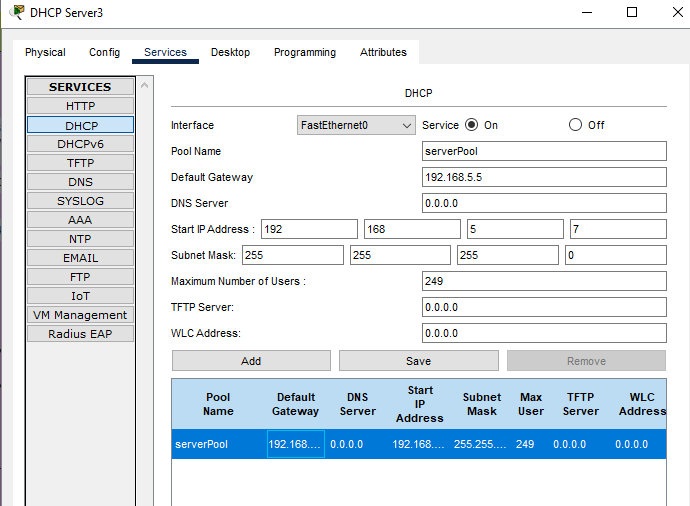
*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*



Configuring the DHCP Server1 pool to start at 192.168.3.7, and configuring the default gateway given to clients, 192.168.3.5.



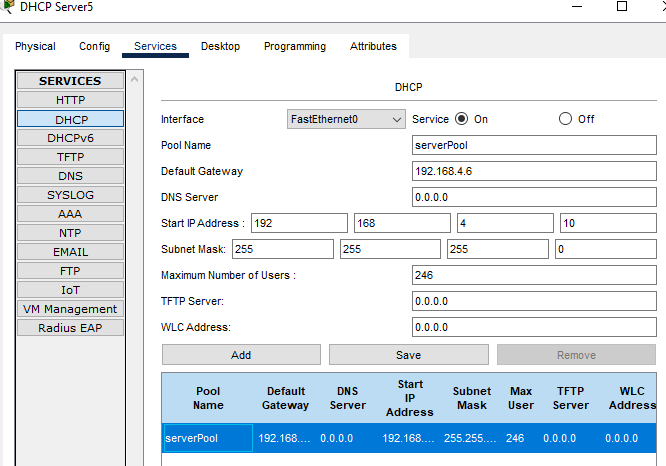
Configuring DHCPServer2



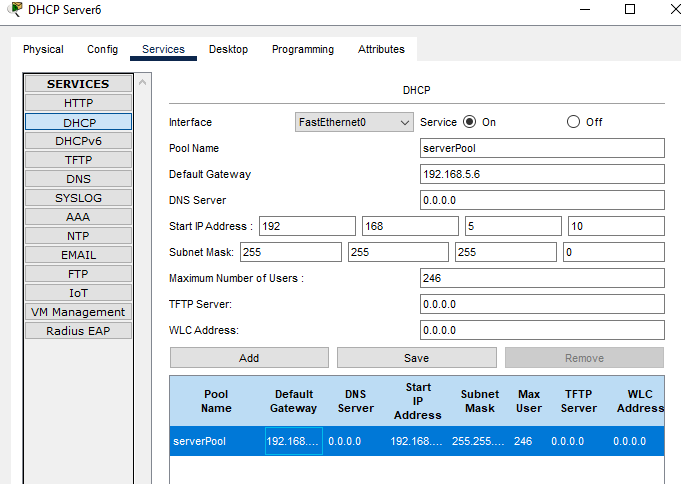
Configuring DHCPServer3



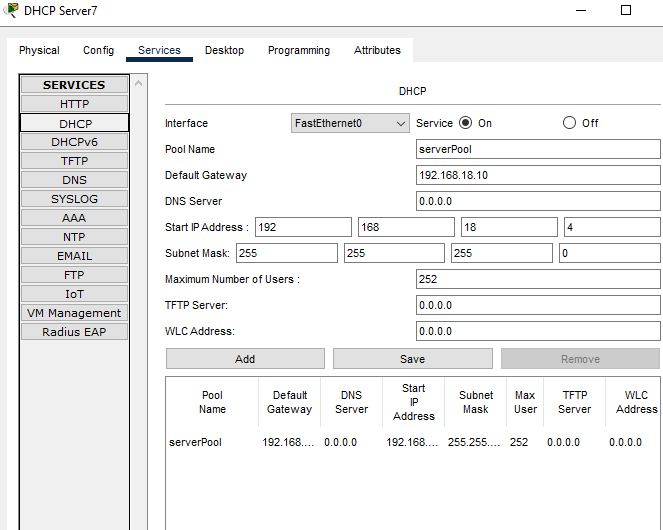
Configuring DHCPServer4



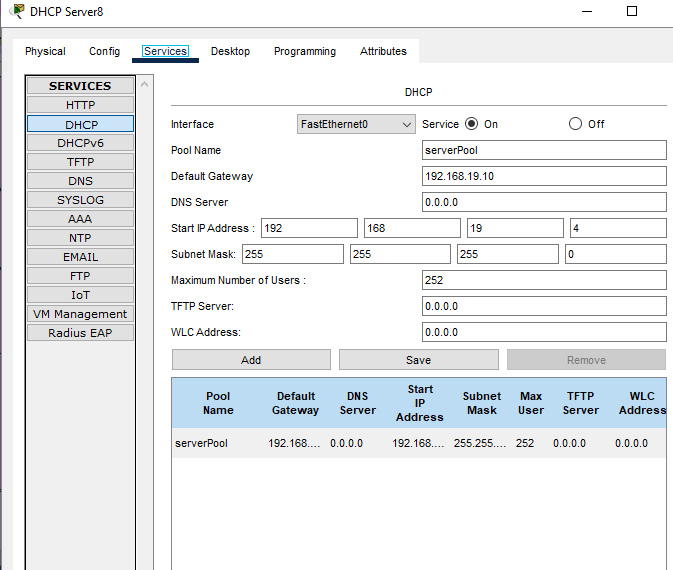
Configuring DHCPServer5



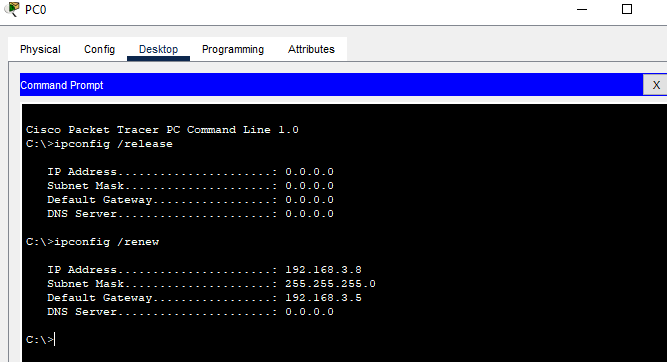
Configuring DHCPServer6



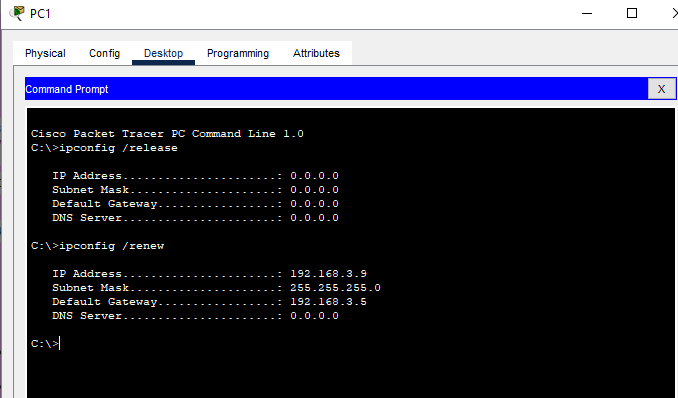
Configuring DHCPServer7



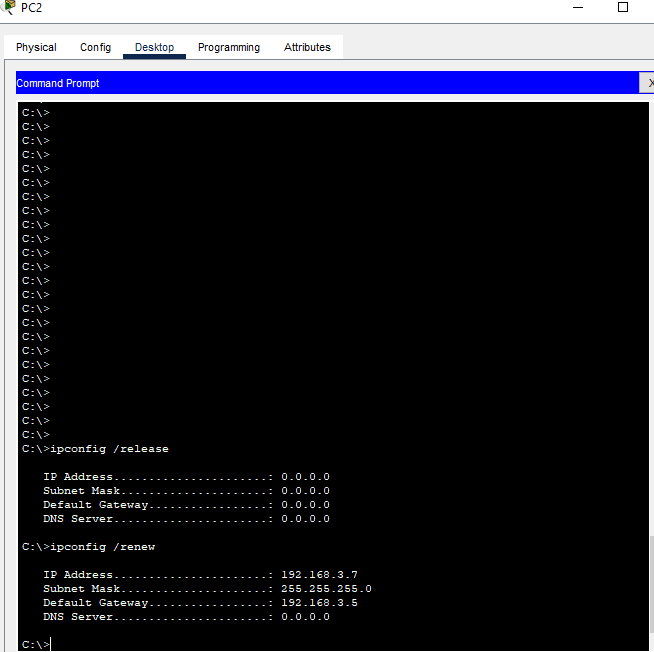
Configuring DHCPServer8



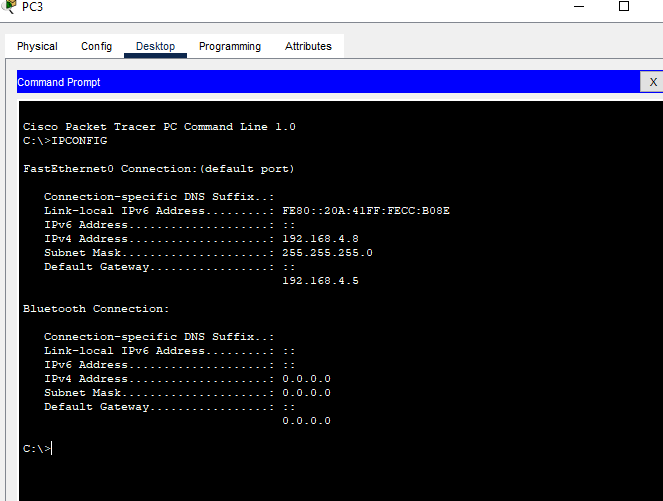
PC0 has successfully acquired an IP address, subnet mask, and default gateway from the DHCP server. The “ipconfig /release” command releases the IP back to the DHCP pool so that another host can use it. The “ipconfig /renew” command starts the DHCP DORA process to acquire an IP address.



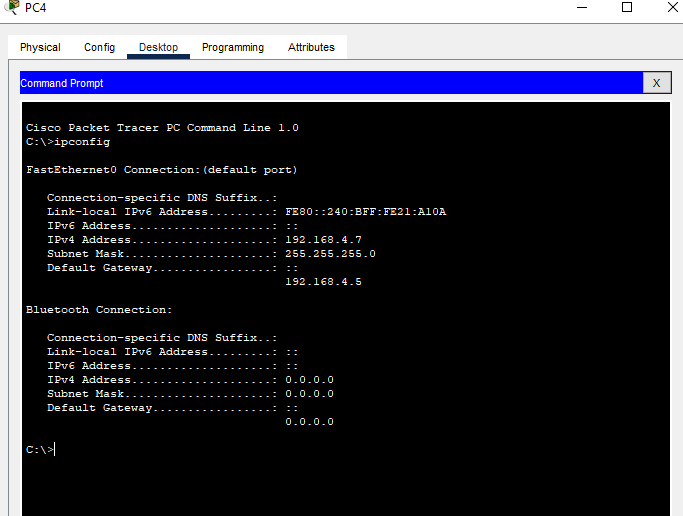
PC1 acquiring an address from the DHCP server



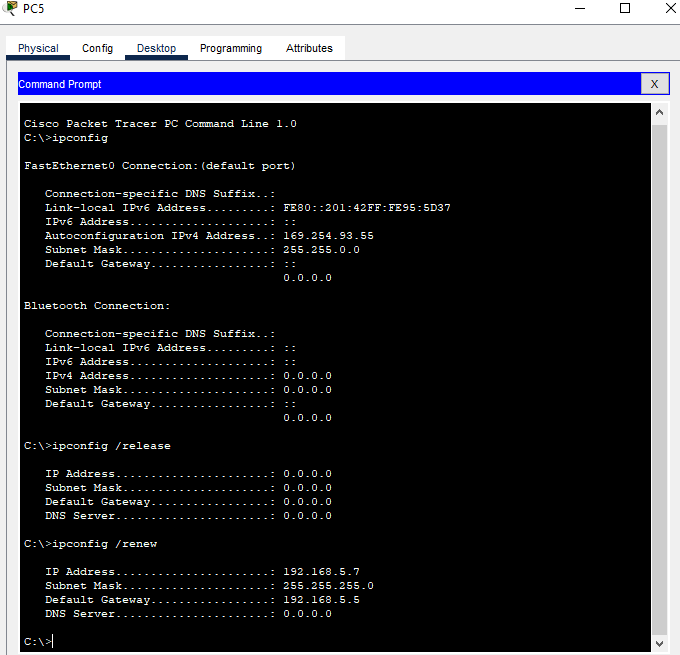
PC2 acquiring an address from the DHCP server



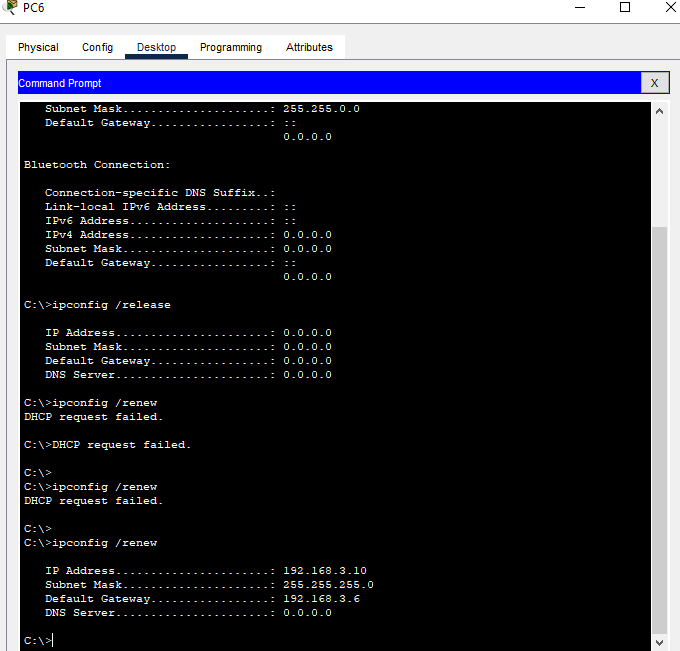
PC3 acquiring an address from the DHCP server



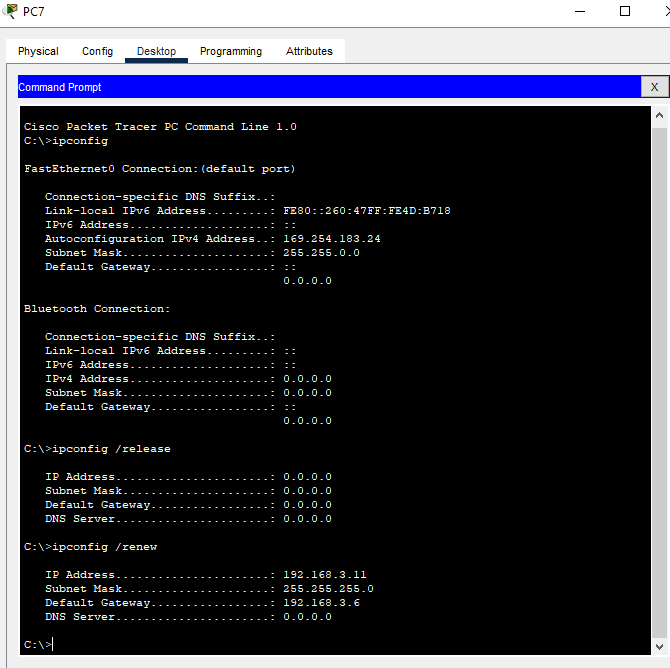
PC4 acquiring an address from the DHCP server



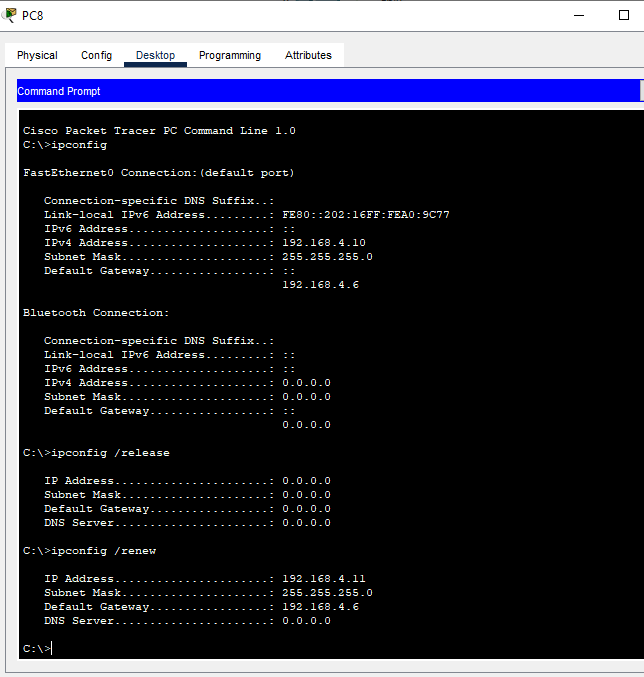
PC5 acquiring an address from the DHCP server



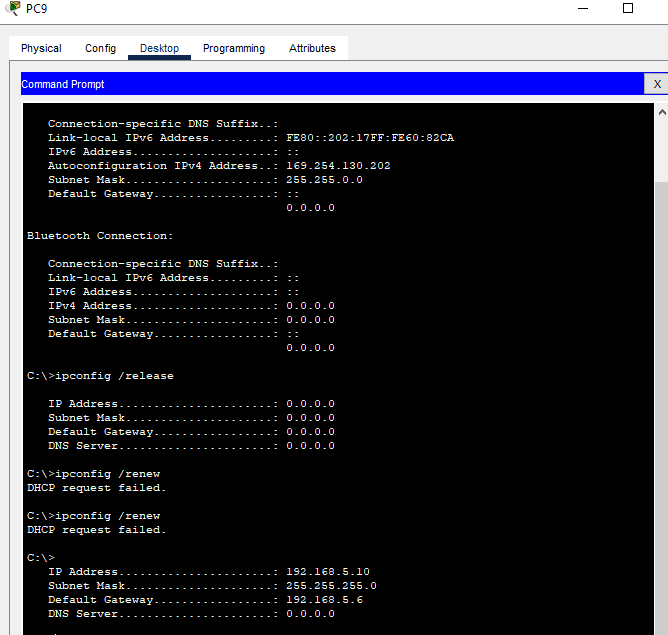
PC6 acquiring an address from the DHCP server



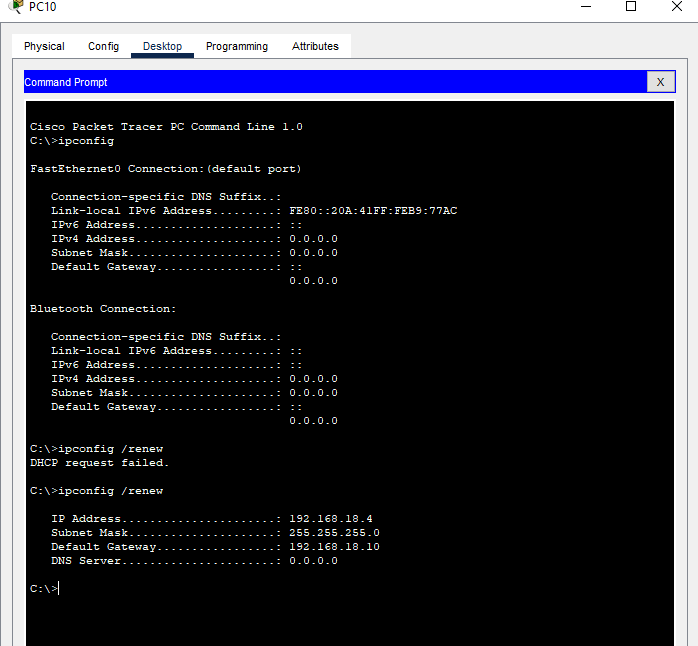
PC7 acquiring an address from the DHCP server



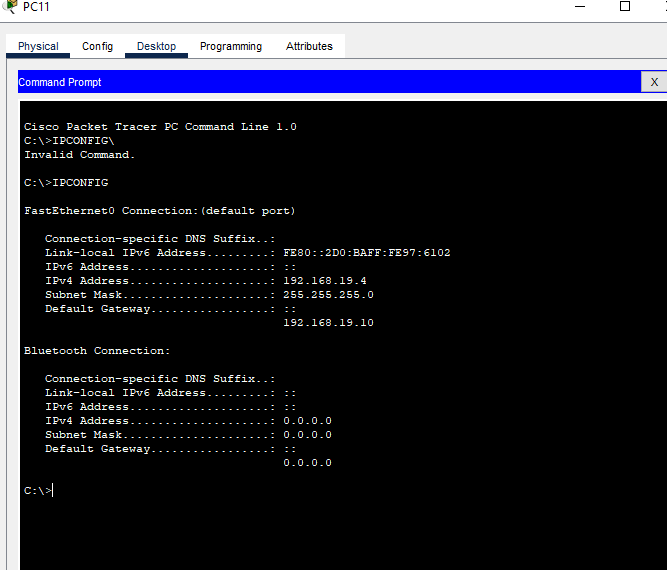
PC8 acquiring an address from the DHCP server



PC9 acquiring an address from the DHCP server



PC10 acquiring an address from the DHCP server



PC11 acquiring an address from the DHCP server

# Test Case #5: Layer 2 Link Redundancy and Spanning Tree Protocol (802.1w)

*Enable and manage the Spanning Tree Protocol to establish redundant Layer 2 paths while avoiding possible loops and broadcast storms. Identify the Layer 2 devices that will become the root bridge.*

## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The functionality of this test case is to prevent the network from experiencing layer 2 loops. This is accomplished by the use of STP: Spanning Tree Protocol. The enhanced version, Rapid STP(RSTP) is the more common version. Actually, Cisco switches use a version of STP called Per-VLAN Spanning Tree(PVST), which is being used in this demonstration. STP accomplishes a loop-free layer 2 network by blocking certain switch ports, preventing broadcast storms from occurring. By default, switches automatically run STP, so there is no need to apply any configurations. Relevant devices include SW1, SW2, SW3, SW4, SW5, SW6, and L3SW1 and L3SW2’s switchports. Relevant tools include the “show spanning-tree” and “show version” commands.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*

A screenshot of a computer

Description automatically generated

A computer screen shot of a diagram

Description automatically generated

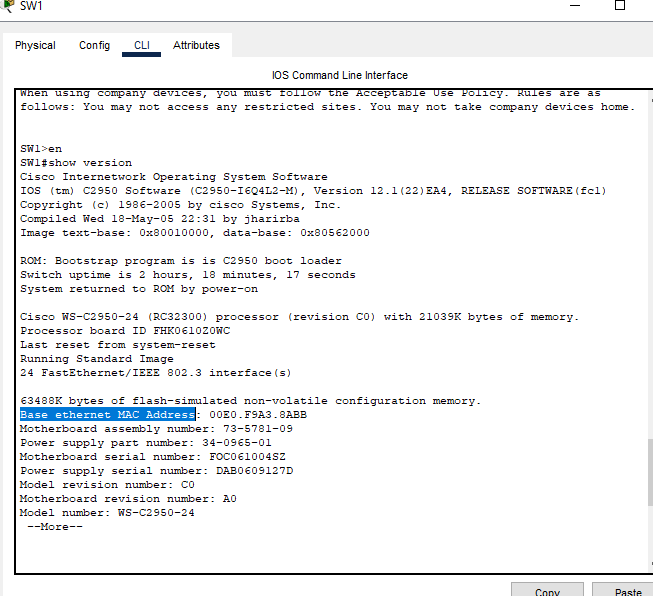
## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

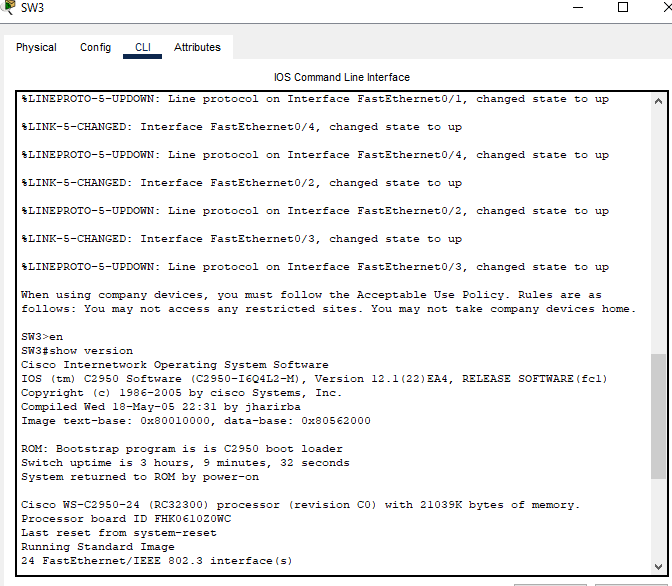
 The testing method I used to verify the root bridge selection process was to use the “show version” and “show spanning-tree” commands. Show-spanning tree will allow you to see which switch is the root bridge. If a switch is the root bridge, it will say directly “this bridge is the root bridge”. To verify that the election process was correct, I used the show version command to see each switch’s base MAC address. I then verified that the root bridge’s MAC was indeed the lowest, which means that the root bridge election process was carried out correctly.

## Process List

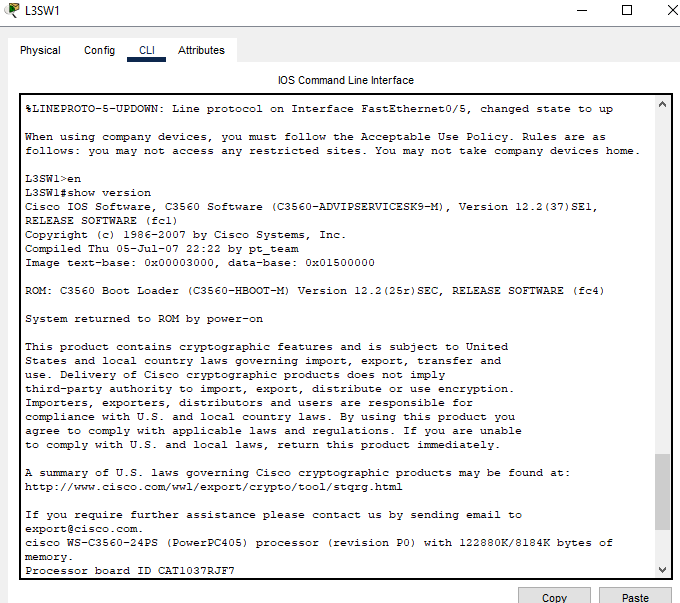
*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*



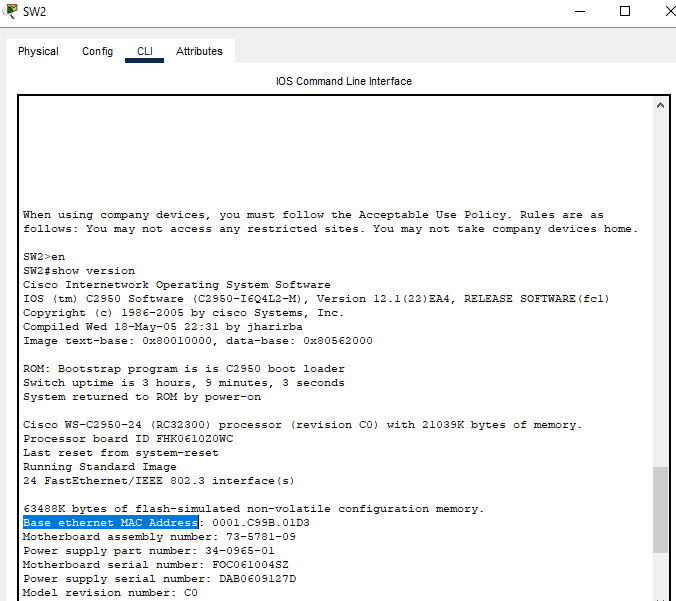
SW1’s MAC address. The “base ethernet MAC address” is the actual device’s MAC address, not the address of any single port on the switch.



SW3’s MAC address



L3SW1’s MAC address



A screenshot of a computer

Description automatically generated

SW2’s MAC address. As we can see by the results of “show version” and “show spanning-tree”, SW2 has the lowest MAC address, therefore it wins the root bridge election.

A screenshot of a computer

Description automatically generated

SW4’s MAC address

A screenshot of a computer

Description automatically generated

As shown by “show version” and “show spanning-tree”, this switch has the lowest mac address, and it is the root bridge for Fenrir Site 2.

A screenshot of a computer

Description automatically generated

SW5’s MAC address

A screenshot of a computer

Description automatically generated

SW6’s MAC address

A screenshot of a computer

Description automatically generated

L3SW3’S MAC address

# Test Case #6: Edge Device Syslog and NTP

*Configure perimeter devices to generate system logs that capture unwanted traffic. Additionally, those perimeter devices should utilize Network Time Protocol (NTP) for clock synchronization.*

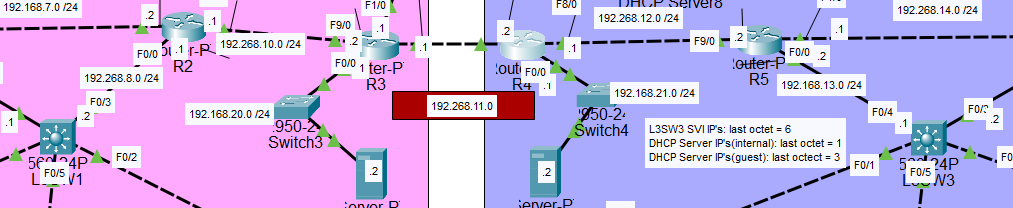
## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The functionality of this test case pertains to first configuring NTP, so that time can be synchronized across devices. This is important for then configuring Syslog, which needs the correct time in order to have accurate logs. Accurate logs are helpful, for example, in an investigation in which you are trying to correlate events and the sequence in which they occurred.

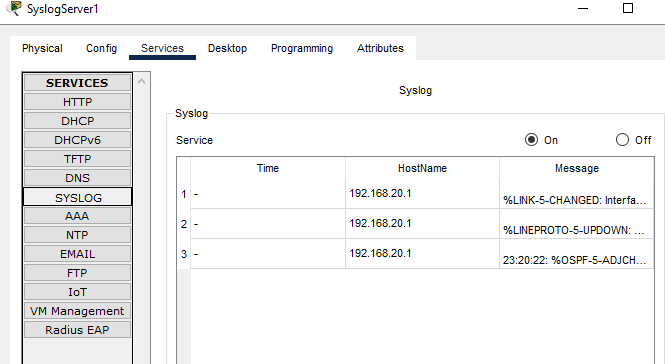
## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*

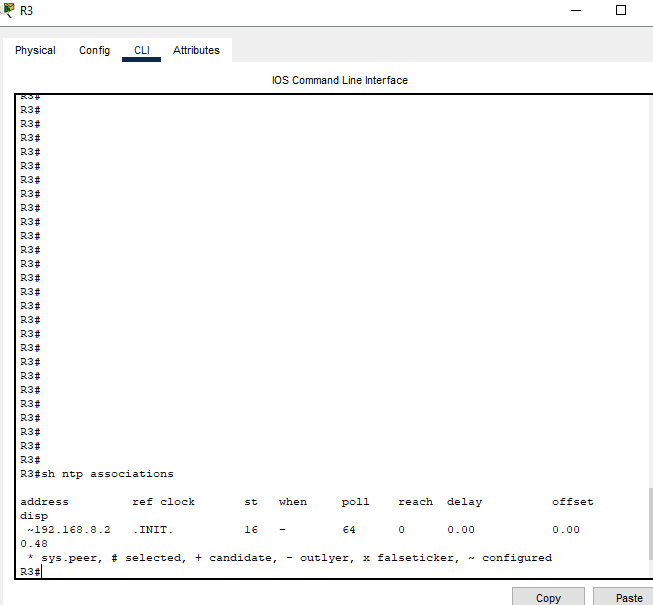


## Testing Method

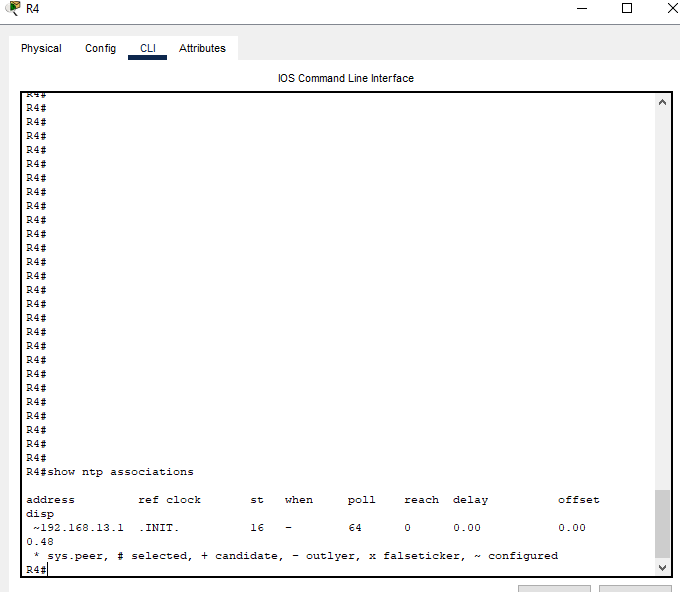
*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*



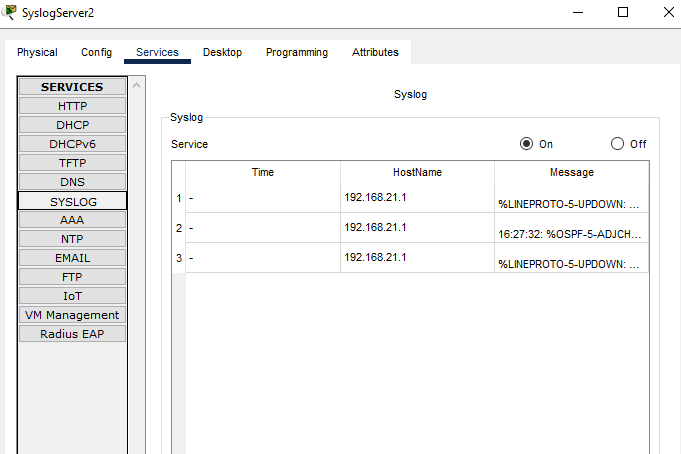
I shutdown an interface on R3 to verify that logging to the external server is working. This is not a great idea for a real network, but for our purposes, it shows that the configuration is correct, and SyslogServer1 is working as intended.



Here I use the show ntp associations command to verify that R3 has associated with its NTP server 192.168.8.2, L3SW1.



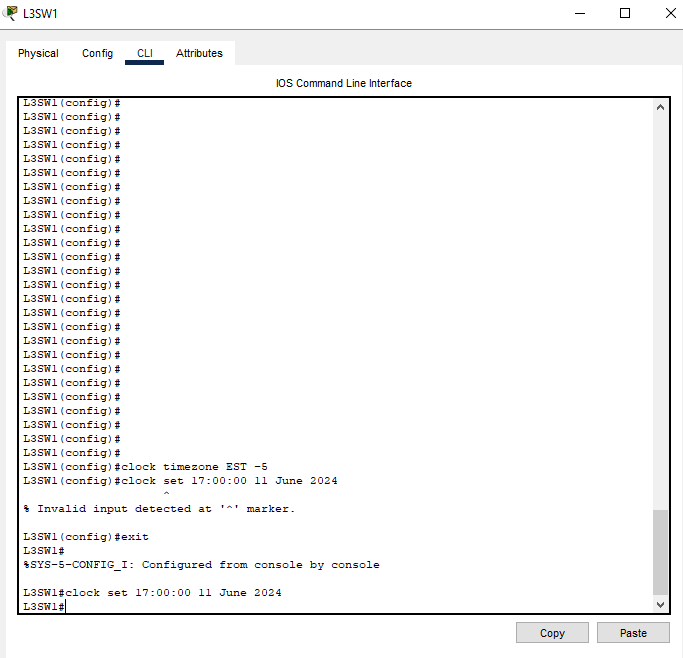
Verifying that R4 has associated with its NTP server 192.168.13.1, L3SW3.



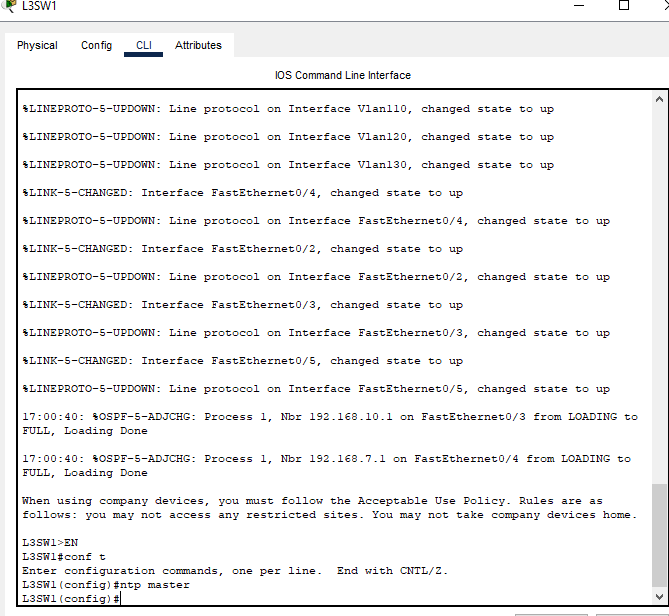
I did the same test for SyslogServer2 as I did for SyslogServer1. Logs are being sent to the external server. The server is functioning as intended.

## Process List

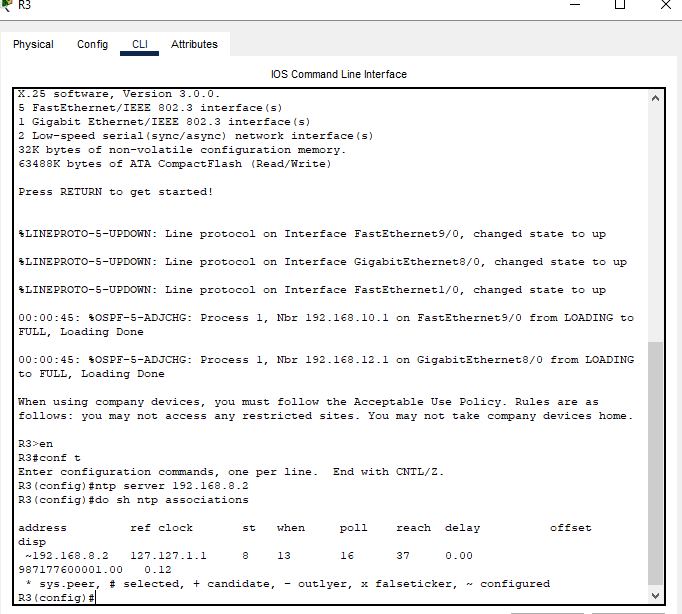
*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*



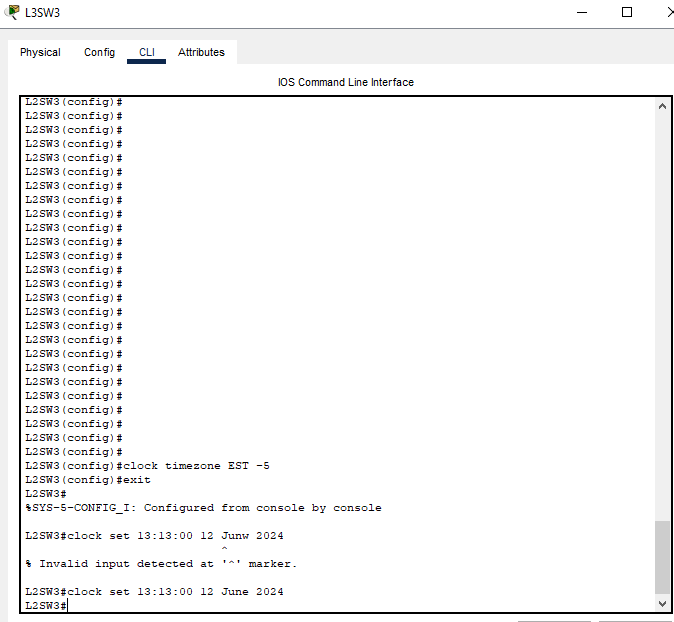
It is important to configure NTP before Syslog so that time across the network will be consistent in the logs. First, I configure the timezone on L3SW1, which will be our NTP server. With the command “clock timezone [timezone] and the offset -5, meaning the time will be 5 hours behind UTC, the default timezone on Cisco devices. We want our NTP server to have accurate time, that is crucial for NTP to work correctly. If the server time is wrong, then all client devices using that server for time synchronization will be wrong too. Using the “clock set” command will allow you to set the device’s software clock time.



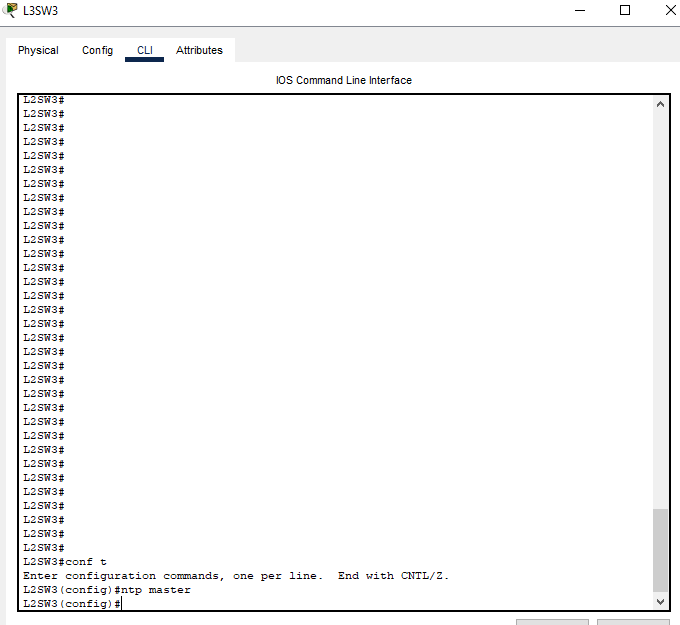
Configuring L3SW1 to become the NTP server for Fenrir Site 1. The “ntp master” command configures the device to act as an NTP server with the default stratum being 8. Other devices can now sync to this server with the command “ntp server [ip]”.



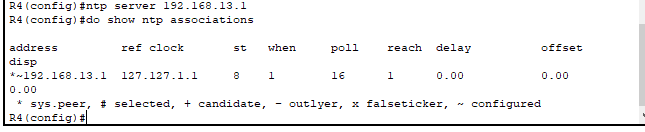
Configuring Fenrir Site 1’s edge router, R3, to use L3SW1 as its NTP server. The IP 192.168.8.2 is L3SW1’s F0/3 interface. The command “show ntp associations” will do just that, it will show which NTP servers that R3 has associated with. If necessary, several NTP servers could be configured as peers to take the place of L3SW1 if it went down.



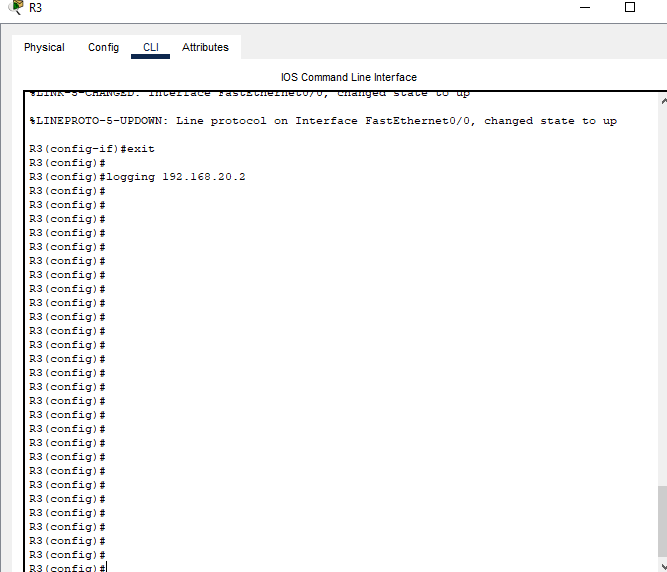
Configuring clock and setting timezone on L3SW3.



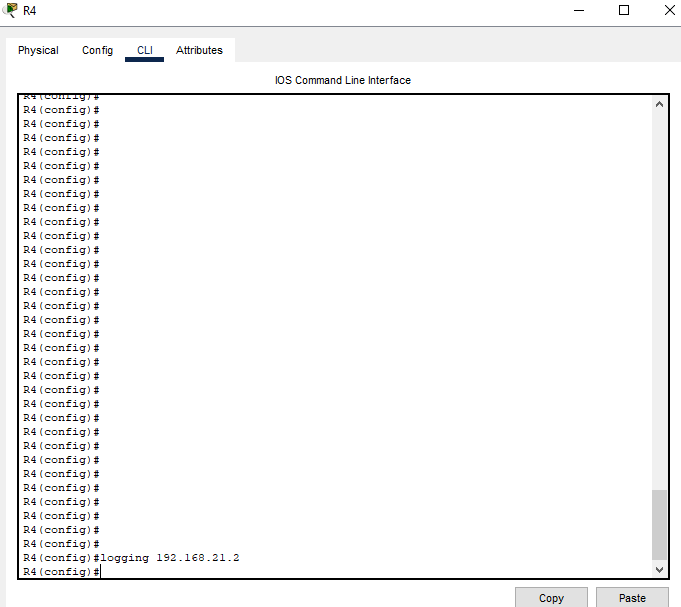
Configuring L3SW3 to act as Fenrir Site 2’s NTP server.



Configuring R4 to use L3SW3 as its NTP server. End of NTP configuration.



Start of syslog configuration. The command “logging [ip]” will configure the device to send its syslogs to an external server, the IP specified in the command. I configure R3 to send them to Fenrir Site 1’s syslog server, SyslogServer1.



Configuring R4 to send its sylogs to an external server 192.168.21.2, which is Fenrir Site 2’s syslog server, SyslogServer2.

# Test Case #7: Basic Network Segmentation at Layer 2 via VLANs and 802.1q

*Your network traffic should be segmented per department or service function at Layer 2 to enhance security and reduce network congestion at the switching layer while allowing segmented traffic to traverse between switches (VLAN trunking).*

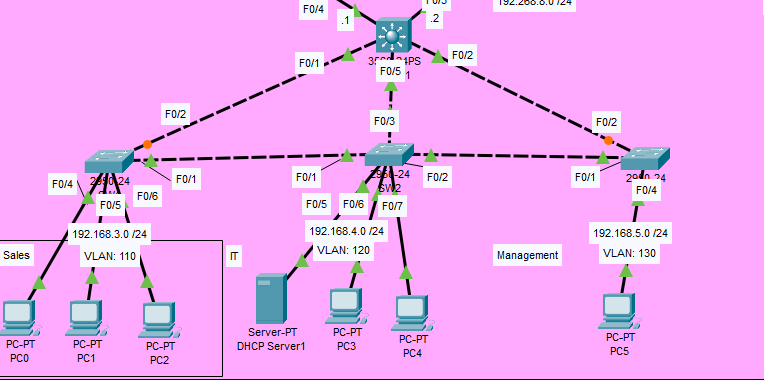
## Functionality

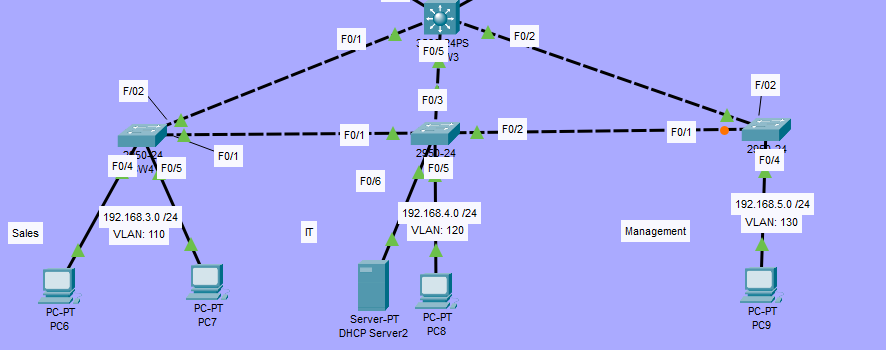
*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The functionality of this test case involves configuring VLAN’s on the L2 switches(access and trunk ports), configuring VLAN’s and SVI’s on the L3 switches, and verifying proper functionality. Relevant tools include the L2 and L3 switches, VLAN 110, 120, 130, subnets 192.168.3.0 /24, 192.168.4.0 /24, 192.168.5.0 /24, and their respective end hosts to verify connectivity.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*





## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

The testing method I used below in the Process List was a combination of the ping and the tracert commands. Ping was used to simply test connectivity between devices. Tracert was used to test the functionality of VLAN’s, and because of inter-VLAN routing, tracert was able to verify that the first hop was the originating device’s default gateway, and not the destination device.

## Process List

*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*

A black text on a white background

Description automatically generated

Creating and assigning VLAN 110 to interfaces f0/4, f0/5, and f0/6 on SW1. Instead of one at a time, I use the “range” command to do multiple. The “switchport mode access” command makes the interfaces access ports, not trunk ports. “Switchport access vlan [number] assigns the interfaces to the VLAN. However, if the specified VLAN does not already exist on the switch, this command will also create the specified VLAN for you.

A black text on a white background

Description automatically generated

Here I am doing the same configuration on SW2 for interfaces f0/5, f0/6, and f0/7. This configuration is for VLAN 120.

A close up of a message

Description automatically generated

SW3 configuration for VLAN 130

A close up of text

Description automatically generated

SW4 configuration for VLAN 110

A close-up of a computer screen

Description automatically generated

SW5 configuration for VLAN 120



SW6 configuration for VLAN 130

Configuration of trunk ports



A group of black text

Description automatically generated

Configuring a trunk port on SW1 f0/1. “Switchport mode trunk” works the same as “switchport mode access” only now for a trunk port. I also configure the allowed VLAN’s, including 110, 120, and 130.

Configuring which VLAN’s are allowed on the trunk. NOTE: In the screenshots there is no “add” in the command, ex. “switchport trunk allowed vlan add 110”. I have realized and fixed this in the actual configuration. Below there will be the output of “show interfaces trunk” on each switch to verify that VLAN’s 110, 120, and 130 are all allowed.

A black text on a white background

Description automatically generated

SW2 trunk port configuration on interface f0/1

A black text on a white background

Description automatically generated

SW2 trunk port configuration on interface f0/2

A black text on a white background

Description automatically generated

SW3 trunk port configuration on interface f0/1

A black text on a white background

Description automatically generated

SW4 trunk port configuration on interface f0/1

A black text on a white background

Description automatically generated

SW5 trunk port configuration on interface f0/1

A black text on a white background

Description automatically generated

SW2 trunk port configuration on interface f0/2

A black text on a white background

Description automatically generated

SW6 trunk port configuration on interface f0/1

A black numbers on a white background

Description automatically generated

L3SW1 SVI configuration for VLAN 110. To create an SVI, I used the interface vlan[number] command, followed by giving the SVI an IP address, and turning the interface on by using the “no shut” command.

A black text on a white background

Description automatically generated

L3SW1 SVI configuration for VLAN 120

A black and white text

Description automatically generated

L3SW1 SVI configuration for VLAN 130

Acc and trunk ports successfully configured.

Configure site 2 SVI’s HERE:

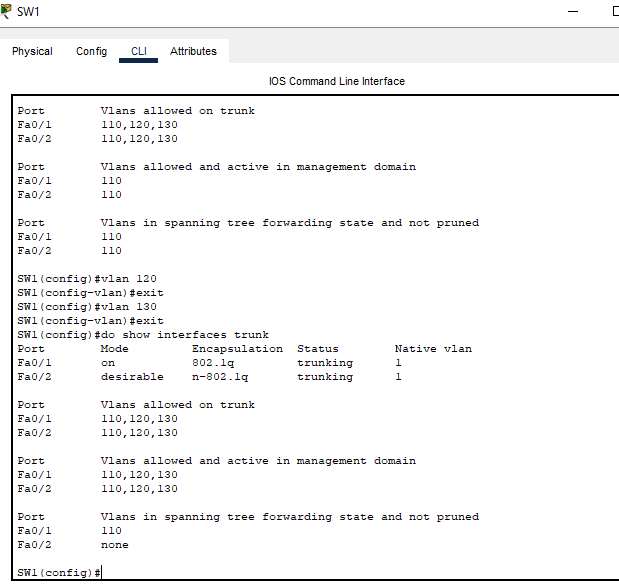


Configuring an SVI for VLAN 110 on L3SW3



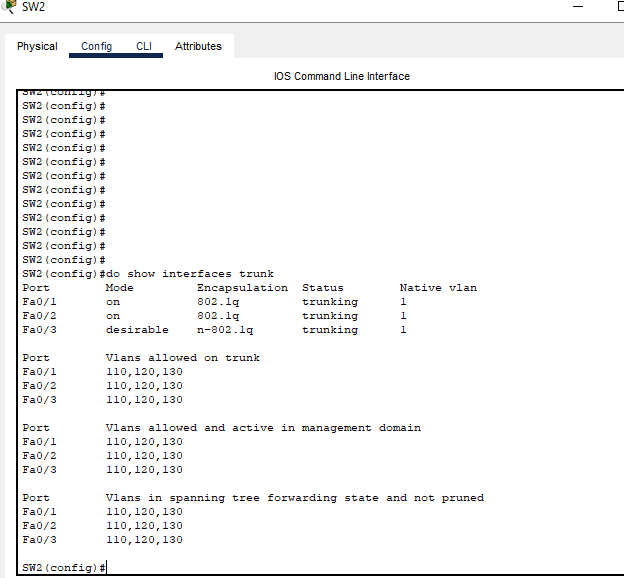
Configuring an SVI for VLAN 120 on L3SW3

  
Configuring an SVI for VLAN 130 on L3SW3

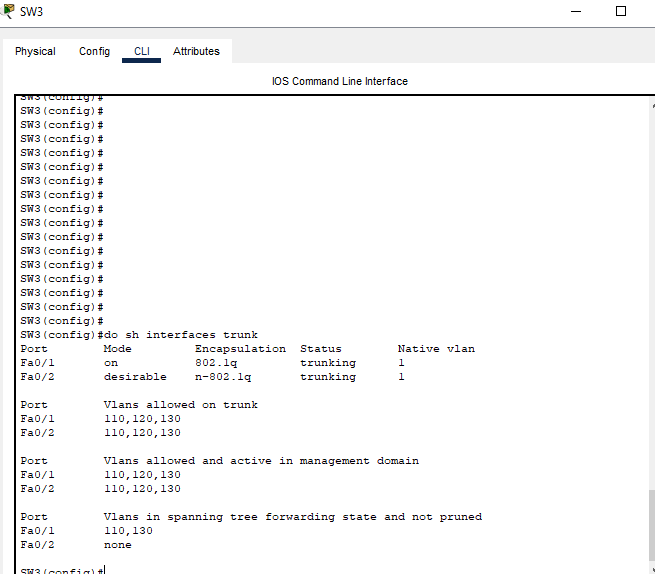


From the earlier note:

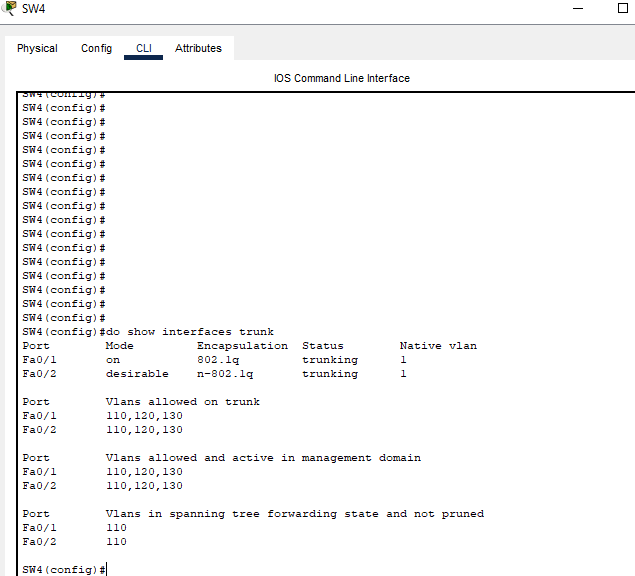
SW1 “show interfaces trunk”



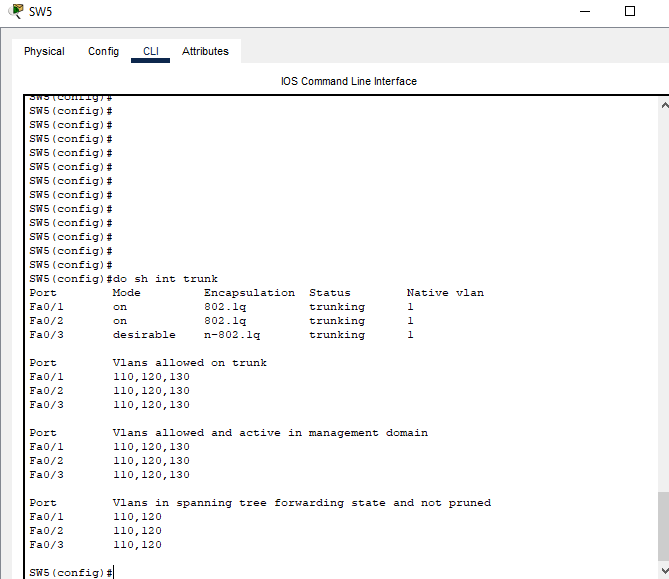
SW2 “show interfaces trunk”



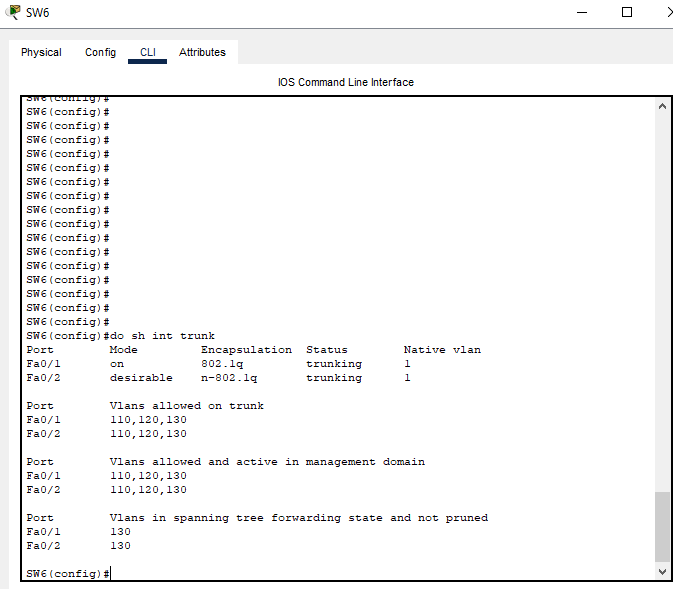
SW3 “show interfaces trunk”



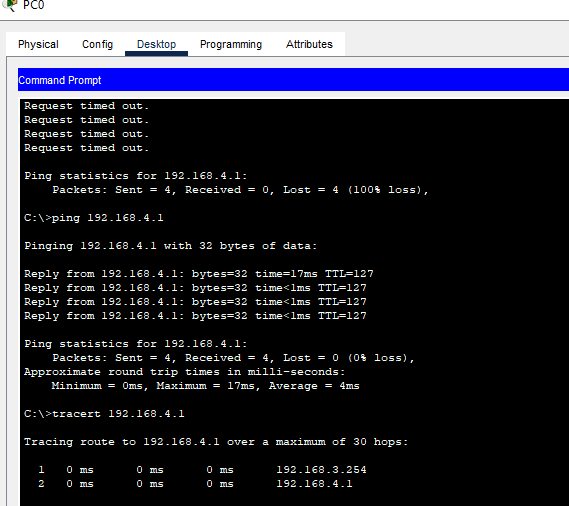
SW4 “show interfaces trunk”



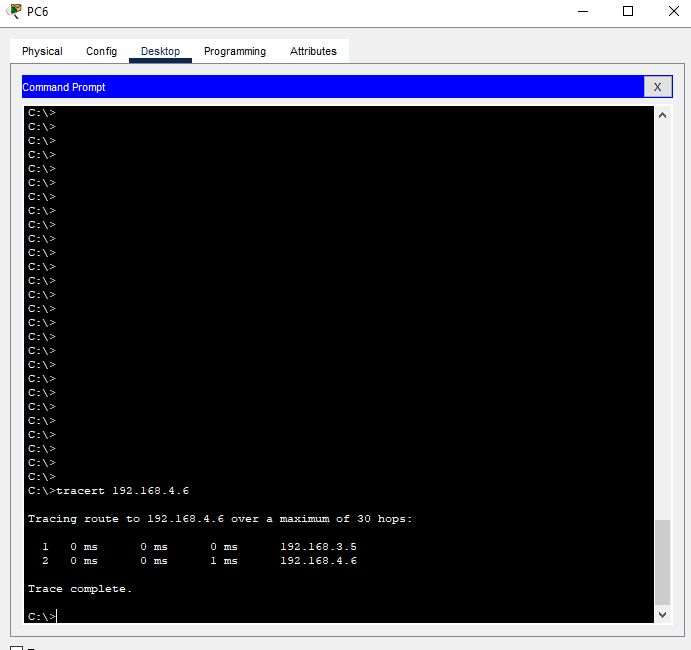
SW5 “show interfaces trunk”



SW6 “show interfaces trunk”



The thing to note here is the tracert command that I ran with the destination of the DHCP server at 192.168.4.1. We can see that the first hop is from PC0 to it’s default gateway, 192.168.3.254, and then to the DHCP server at 192.168.4.1. If the DHCP server were in the same broadcast domain, that first hop to the default gateway would not be needed, since they would be in the same subnet. Since I have separated the LAN into different VLAN’s, inter-VLAN routing is needed, which is why traffic first went to PC0’s default gateway. This shows that the VLAN’s are operational and working as intended for Fenrir Site 1.



The tracert from PC6 to DHCP Server2 at 192.168.4.6 goes through PC6’s default gateway first, just like from PC0 to DHCP Server1, proving that the VLAN/SVI configuration is functioning properly.

# Test Case #8: Basic or Advanced Networking - OSPF

## Custom Test Case

*Define a* ***custom test case*** *to be run within your network project aligned to your specific organizational need or opportunity identified in Task 1. The custom test case should be equivalent in scope and requirements to the predefined test cases and pertain to basic or advanced networking.*

## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The purpose of this test case is to configure OSPF on the routers and L3 switches upstream interfaces for both Fenrir Site 1 and 2, and to verify that routes are being shared, routers are becoming OSPF neighbors, and adjacencies are being formed. Relevant devices include R1, R2, R3, R4, R5, R6, L3SW1, and L3SW2. Relevant subnets include 192.168.6.0 /24, 192.168.7.0 /24, 192.168.8.0 /24, 192.168.10.0 /24, 192.168.12.0 /24, 192.168.13.0 /24, 192.168.14.0 /24, and 192.168.15.0 /24.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*

A computer screen shot of a computer

Description automatically generated

## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

The testing method I used to verify functionality were several commands. These include the “show ip ospf neighbor” command, and the “show ip route” command. The command “show ip ospf neighbor” will show the devices OSPF neighbors, current dead timers, and also the interface. For more granularity, you can use the “show ip ospf interface [interface]” command. This will show you information such as network type, neighbor count, whether the device is designated, etc. I used the “show ip route” command to verify that OSPF routes were actually being shared between neighbors. These commands are fairly simple but allow for easy verification that OSPF, and dynamic routing in general, is functional.

## Process List

*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*

A screenshot of a computer

Description automatically generated

Configuring OSPF first on L3SW1. From global configuration mode, you can use the “router ospf [process ID]” command to enter OSPF configuration mode. The process ID is entirely local, unlike EIGRP, in which they would have to match in order for routers to become neighbors. OSPF routers however, can have different process ID’s and still become neighbors. The network command’s syntax is as such: “network [matching interface IP that you want to activate OSPF on] [wildcard mask] area [area]. Since I’m configuring single area OSPF, it’s easiest to use area 0. This command specifies which interface you want to activate OSPF on, not which network. The wildcard mask is essentially an inverse of a subnet mask, meaning a /24, like which I used here, would be 0.0.0.255 instead of 255.255.255.0. Once I configure OSPF on a neighboring router, you can expect to see a Syslog message indicating they have become neighbors, as long as the configuration is correct and matching(hello and dead timers, unique router ID’s, etc).

A screenshot of a computer

Description automatically generated

R1 OSPF configuration. R1 has not become neighbors with R2, as I have not configured OSPF on R2 yet. R1 and L3SW1 however, are neighbors.

A screenshot of a computer

Description automatically generated

R2 OSPF configuration. Now, R2 and R1 have become neighbors.

A screenshot of a computer

Description automatically generated

R3 OSPF configuration. R3 is neighbors with R2, but not yet with R4, as R4 has not been configured yet.

A screenshot of a computer

Description automatically generated

R4 OSPF configuration

A screenshot of a computer

Description automatically generated

R5 OSPF configuration. R5 is neighbors with R4, but not yet with R6, as R6 has not been configured yet.

A screenshot of a computer

Description automatically generated

R6 OSPF configuration

A screenshot of a computer

Description automatically generated

L3SW3 OSPF configuration

# Test Case #9: IPSec

## Custom Test Case

*Define a* ***custom test case*** *to be run within your network project aligned to your specific organizational need or opportunity identified in Task 1. The custom test case should be equivalent in scope and requirements to the predefined test cases and pertain to IPSec.*

## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

The functionality of this test case is to ensure that communications between both Fenrir sites is encrypted and secure. This will be accomplished with the use of an IPsec site-to-site VPN tunnel. Relevant devices include Router0, Router1, and the 192.168.10.0 /24, 192.168.11.0 /24, and 192.168.12.0 /24 subnets.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*

A green and black rectangular with black text

Description automatically generated

## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

A screenshot of a computer

Description automatically generated

With the “show crypto ipsec sa” command, I was able to verify the functionality of the IPsec tunnel on Router0. Pings were sent across the network beforehand to populate the fields shown in the command output. What’s important here is the “pkts encrypt” field, which shows a counter of the packets which were encrypted and sent over the tunnel. This means that the site-to-site IPsec VPN between Fenrir Site 1 and 2 is working as intended.

A screenshot of a computer

Description automatically generated

Here’s the output of the same command, only this time on Router1. Packets are being encrypted, and the tunnel is working as intended.

## Process List

*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*

After attempting to configure an IPsec VPN between R3 and R4, I realized that the routers do not have the capability to acquire the security license necessary for the configuration. Therefore, instead of potentially creating unwanted problems in the network, I will create another small network to showcase the IPsec VPN configuration.

A screenshot of a computer

Description automatically generated

Configuring the security license on Router0. A security license is required for many security configurations, such as IPsec and ISAKMP. Therefore, it needs to be activated first. The license is acquired, and then the router is reloaded to activate the license.

A screenshot of a computer

Description automatically generated

Configuring the security license on Router1.

A screenshot of a computer

Description automatically generated

Configuring an ACL on Router0 for the tunnel between Router0 and Router1. The ACL determines which traffic will actually be encrypted and sent over the IPsec tunnel.

A screenshot of a computer

Description automatically generated

Configuring an ACL on Router1. This is for the tunnel Router1 will have established with Router0.

A screenshot of a computer

Description automatically generated

Configuring the ISAKMP policy on Router0. This enables the router to negotiate with another router on how they want to establish an IPsec association. Here I created the policy numbered 10, configured 256 bit AES encryption, pre-shared key authentication, and configured it to use group 5 for key exchange using Diffie-Hellman.

A screenshot of a computer

Description automatically generated

Configuring the ISAKMP policy on Router1. The same configuration used on Router0 is also applied to Router1.

A screenshot of a computer

Description automatically generated

Here I configured the pre-shared key to be “cisco”, and configured the peer address to be Router1’s outside interface.

A screenshot of a computer

Description automatically generated

I applied the same configuration on Router1, only now with Router0’s outside interface as the peer.

A screenshot of a computer

Description automatically generated

Configuring the transform set on Router0 to use ESP with 256 bit AES, and ESP-SHA with HMAC. I have named the transform set “Router->Router1”.

A screenshot of a computer

Description automatically generated

Configuring the transform set on Router1. The configuration is the same for Router1, only now I have named it to be “Router1>Router0”.

A screenshot of a computer

Description automatically generated

Configuring the crypto map on Router0. It is named “fenrir-map” and the peer has been set to Router1’s outside interface. Group5 Diffie-Hellman is used again with PFS, and the default security association lifetime is configured. Then, the transform set is set to the crypto map, and it will match based on the ACL I configured earlier, ACL 150.

A screenshot of a computer

Description automatically generated

Configuring the crypto map on Router1. The difference here is that the peer has been changed to Router0’s outside interface, and the decription has also been changed to “Router1>Router0” for clarity.

A screenshot of a computer

Description automatically generated

Applying the crypto map on Router0’s outside interface. This is done by going into interface configuration mode of the router’s outside interface; the one we are establishing a tunnel on(G0/0), and using the command “crypto map [map]”.

A screenshot of a computer

Description automatically generated

Applying the crypto map on Router1's outside interface. It has beena applied on Router1’s G0/0 interface.

# Test Case #10: Network Security

## Custom Test Case

*Define a* ***custom test case*** *to be run within your network project aligned to your specific organizational need or opportunity identified in Task 1. The custom test case should be equivalent in scope and requirements to the predefined test cases and pertain to network security.*

## Functionality

*Describe the* ***functionality*** *of the test case in relation to your network project. Identify the relevant tools (devices, subnets, etc.) used in this test case and their specific interactions.*

This test case will be dedicated to the functionality of Port Security. For security purposes, there should be some default behavior of a network device when someone plugs their end device into it. For the purposes of my network, I will use the default amount of allowed MAC addresses for port security, only one. This way, once a device is already connected, an unauthorized user will not be able to plug in their device to a switch and gain access to the network. If they do, the interface will become err-disabled, essentially in a shutdown state.

## Network Diagram or Segment

*Provide a* ***network diagram or segment*** *visualizing the topology and devices used in this test case.*



A close up of numbers

Description automatically generated

## Testing Method

*Summarize the* ***testing method*** *used to verify functionality of the network project within the virtual lab environment, including any metrics of success.*

A screenshot of a computer

Description automatically generated

By using the command “show port-security interface [interface]”, with this interface being f0/4, you can see port security information for that port. PC0 is connected to SW1’s f0/4 interface. I have sent some pings from PC0 beforehand, so the “last source address” field is populated with PC0’s MAC address and source VLAN. Notice that the “maximum MAC addresses” field is 1. So, when I disconnect PC0 from the interface and connect an IP phone, the port status should change from “secure-up” to shutdown, and the last source address field to the new PC’s MAC address. The port-status being shutdown will confirm that port-security is working as intended. Since I did not configure err-disable recovery, the port will not come back online automatically. I will have to manually use the “shutdown” and “no shutdown” commands for it to come back online.

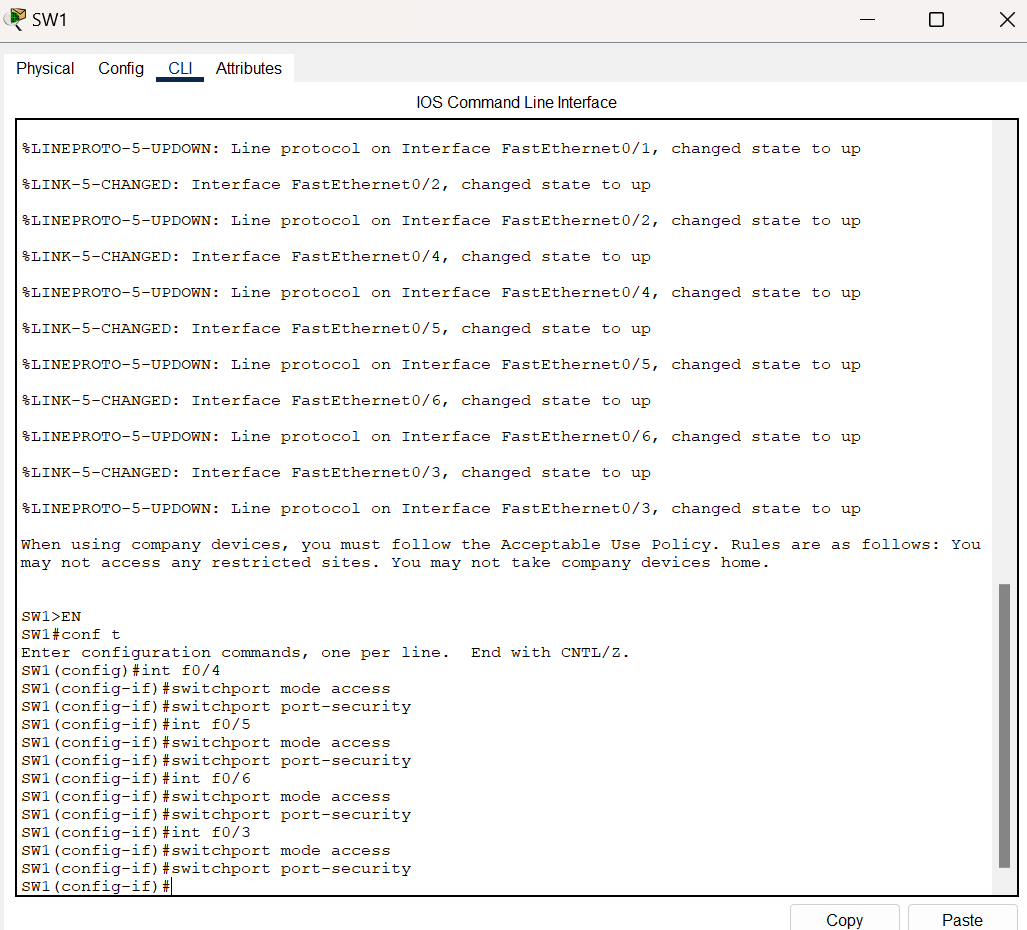
A screenshot of a computer

Description automatically generated

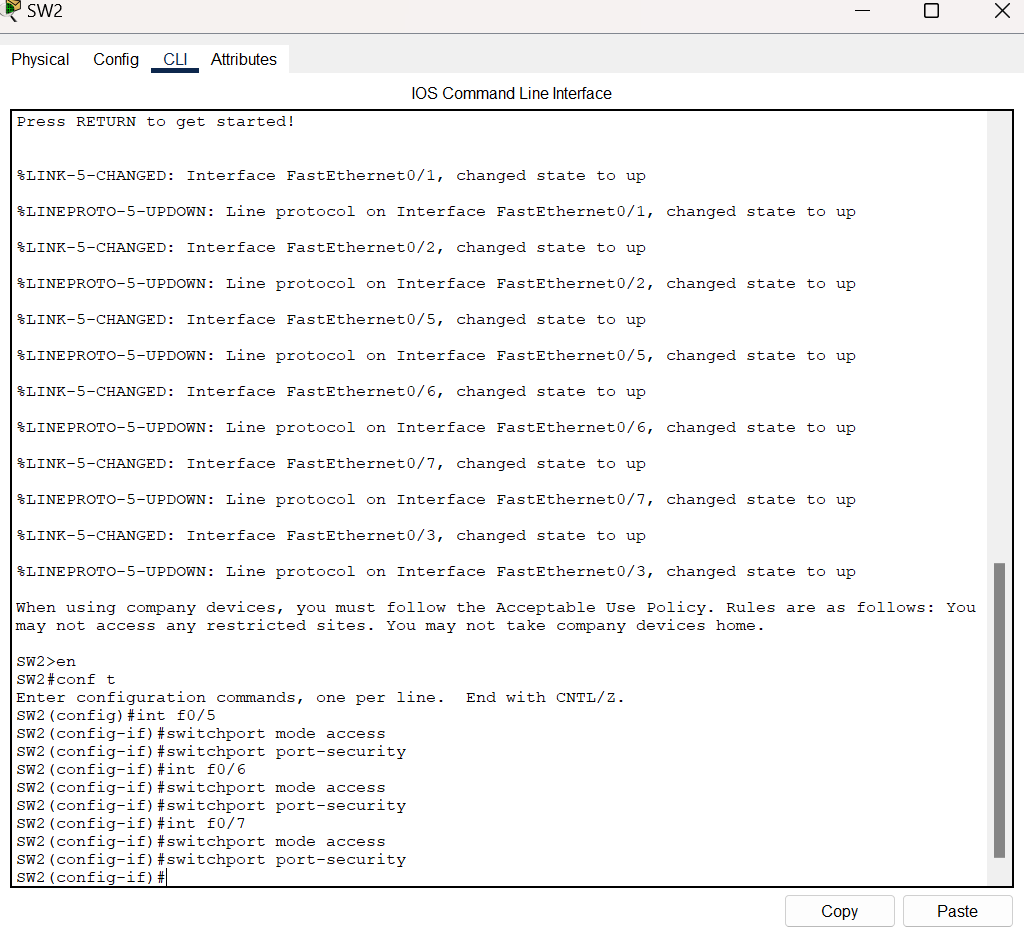
After adding an IP phone to interface f0/4, the port status has changed to “secure-down” and the total MAC addresses field has been cleared. This verifies that port security is functional and working as intended. By err-disabling the port, it will essentially be useless to an attacker attempting to access the network.

## Process List

*Provide a comprehensive* ***process list*** *of the steps taken within the network project to run the testing method. Include screenshots to illustrate the process and ensure clarity for others attempting to replicate the test.*



SW1 Port Security configuration. First, the port has to be configured statically as either access or trunk, it cannot be in a dynamic mode. So, I used the command “switchport mode access” to put the interfaces in a static access mode. Then I configured port security with the default settings by using the command “switchport port-security”. The allowed MAC addresses on each port will be only 1, which is my intent. This way, the interfaces will become err-disabled if the switch learns another MAC address on that port.



SW2 port security configuration

A screenshot of a computer

Description automatically generated

SW3 port security configuration

A screenshot of a computer

Description automatically generated

SW4 port security configuration

A screenshot of a computer

Description automatically generated

SW5 port security configuration

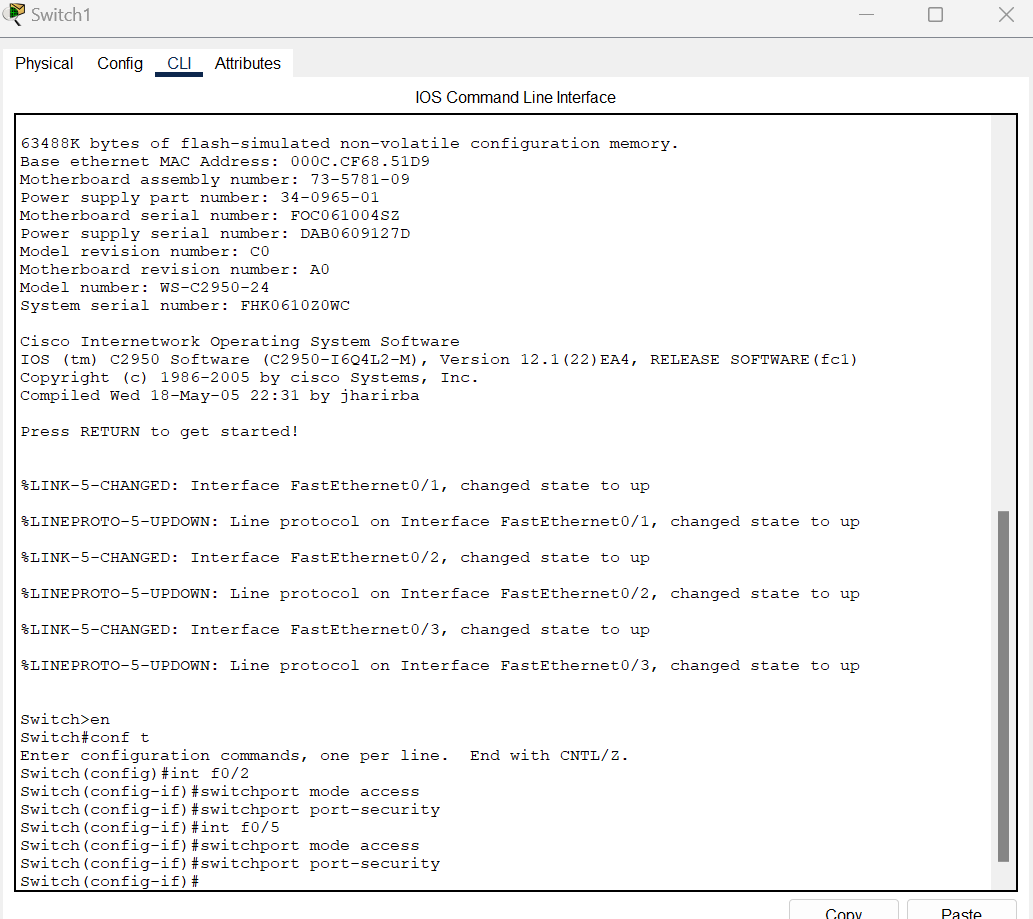
A screenshot of a computer

Description automatically generated

SW6 port security configuration



SW0 guest network port security configuration



SW1 guest network port security configuration

# Network Troubleshooting

*Discuss how you analyzed the network to identify, troubleshoot, and resolve issues during development or when ensuring functionality of the test cases.*

Troubleshooting my network was the major reason why I got stuck so many times. Mostly, it was simple issues that I overlooked. Some were more thought-provoking and took longer to figure out. I take the following steps during my troubleshooting process:

1. Ask myself basic questions, such as “what’s happening? Was that configuration correct?” Questions like those help me to get my mind in troubleshooting mode.
2. Next I try to isolate the problem. If I can trace the problem to a section of the network, it will be easier to diagnose.
3. Here I will check basic configurations, ie. IP/subnet mask, default gateways, routes, is the interface online, etc. Problems in this area were plentiful, and even after I thought everything was correct, I realized that I overlooked a smalld detail.
4. “Show” commands are extremely important for troubleshooting, they allow you to see your configurations up close and in detail. I also use the “?” added onto a command to see all possible configurations. This is where I spend most of my time.
5. If none of these steps resolve the problem, I start the research phase. Cisco forums are especially helpful, as they are vendor specific and there usually is someone who has had the exact same problem before.
6. Lastly, I take breaks. The last thing you want is to get burnt out and tired. Sometimes all you need is to take a break, come back the next day, and tackle the problem with a fresh mind.

Thank you for reading my Network Engineering Capstone project.

**Emily Parker**

**011200197**

**epar255@wgu.edu**