Configure and Verify ACLs in a Network Environment and Configure and Verify IP Services

Your Name: ­­­­­­­­­­­­­­­­­­­­­­­­Shivani Patel

DSI Number: ­­­­­­­­­­­­­­­­­­­­­­­­D40352579

NETW204

DeVry University

Date: ­­­­­­­­­­­­­­­­­­­­­­­­08/14/2015

Professor Name: ­­­­­­­­­­­­­­­­­­­­­­­­Jayson Sayers

Table of Contents

[Introduction 3](#_Toc396564211)

[Procedure 3](#_Toc396564212)

[Results 5](#_Toc396564213)

[Application and Analysis 5](#_Toc396564214)

[Questions 5](#_Toc396564215)

[Conclusion and Recommendation 6](#_Toc396564216)

[Lessons Learned 6](#_Toc396564217)

[References 6](#_Toc396564218)

# Introduction

Access control lists are exactly what the name means, there are security measures that control which devices get access to the network and it defines what their privileges are. To implement an ACL is not too difficult, but to troubleshoot ACLs and make sure the lists are permitting the correct users to gain access to the network, that is another lesson in itself. For this lab we will create ACLs and implement them on the network, then we will learn the methods to troubleshooting issues with ACLs. For the second part of the lab, we will deal with IP services. IP services provide critical network services to help the network administrator deploy, manage, and control IP network. It is necessary to enable IP services on the network and to make sure they are working correctly to manage a functioning network.

# Procedure

This was an extremely dense lab to go through so for the purposes of understanding the material I have every command we used and why we used it written out here.

#access-list is the command used to create lists and name them. Standard lists are numerical and extended lists are numerical as well as named. The function of these lists differs by the depth of the description. For the lab we used list “1” as out starting list.

#access-list 1 permit ? is what we used to name the list and permit it to allow an address. The question mark is used to prompt the administrator the different things we can substitute the “?” with. We were returned with six different options as such:

Hostname or A.B.C.D Address to match

any Any source host

host A single host address

#access-list 11 deny {ip address wild card masks} was used to deny the specific address from accessing the network. Here we used the “deny” command to deny instead of the “permit” command to allow.

The key to a successful access list implementation is to make sure the list exists. That’s why we used the #show ip access-lists command to show all of the access lists that were created. There was also a NAT access list but we were instructed not to do anything to that yet.

Next we needed to take the lists that we created and bind them to an interface based on inbound or outbound traffic. The command #interface gigabitethernet 0/1 accessed interface g0/1, at this poing we used the #ip access-group 10 out command to assign outgoing traffic on that interface the ACL group ten. Then we accessed g0/0 and assigned the ACL to the traffic coming in to the interface. We noticed that because of the NAT router, the placement of the ACL on g0/0 would be futile so we removed the ACL with the #no ip access-group 10 in command on the interface using the #int g0/0 command.

Here there is the configuration of an extended ACL, the extended ACL is more nit picky about its specifications. The access lists we created is ACL 110, we permitted ftp on protocol on select hosts but deny ICMP on those hosts.

#access-list 110 permit tcp host 192.168.16.10 any eq ftp

#access-list 110 permit tcp host 192.168.16.10 any eq ftp-data

#access-list 110 deny icmp host 192.168.16.10 host 172.15.0.10

Use the #show access-lists 110 command to see the access lists you created. After that we got on interface configuration mode for g0/0 and assigned ACL 110 to incoming traffic on that interface.

#interface GigabitEthernet 0/0

#ip access-group 110 in

The next part was creating a named access control list; here we just substituted the numerical name with the alphabetical name of the access list. The name of the list was ftp so to seethe list that we created we needed to specify the named list with the #show ip access-lists permit-ftp command.

For remote access of ACLs it was a matter of creating a named standard or extended list (we used remote as the name) and opening up #line vty 0 15, line vty configuration mode and using the #access-class permit-remote in, command to set ACLs on VTY lines for remote access. If you want only SSH connections then that should be configured in VTY configuration mode prior to enabling access lists on the VTY lines.

For the second part of the lab we created a dhcp pool called labpool and specify the network for that pool. For the command we have to include the mask length instead of subnet mask. Afterwards we have to exclude a high and low address to reserve the addresses in the range. Here we used the “?” after the command to give us options of what to input. It was helpful for learning multiple new commands all at once.

#ip dhcp pool labpool

#network 192.168.16.0 /24

#ip dhcp excluded-address 192.168.16.1 192.168.16.15

After the pool was created we went back to dhcp configuration mode for labpool and configured DNS server (192.168.16.10), default gateway, and a domain name, as well as a lease length of seven days for the pool.

#ip dhcp pool labpool

#dns-server 192.168.16.10

#domain-name practicelabs.com

#default-router 192.168.16.1

#lease 7

To test the configuration we went to NYWAN1 to try to configure an address on g0/0 dynamically. Recall these commands are not from the router but from NYWAN1. After that we went to privileged mode to use a show command to see the dynamically assigned ip address.

#interface gigabitethernet 0/0

#ip address dhcp

#no shutdown

#end

#show ip interface brief

“#show ip dhcp binding” was used to verify dhcp assignments

Troubleshoot with “#show ip dhcp conflict” if you get a conflict error. Other commands included:

#clear ip dhcp binding {binded ip address}

#show ip dhcp binding

#clock set 09:00:00 August 1 2015 (we purposely made it different)

#show clock on NYWAN1 shows that there is a different time on the router

#ntp server 192.168.16.1 was used on NYWAN1 to configure an NTP server on the switch.

#show ntp associations

#show clock

^ These commands are used to verify NTP on the router. We verify that the clock has changed to what we set it as previously on the router.

Syslog was on of those things that were optional on the network, but to configure it the administrator has to go on a device and change the NIC back to the original settings, then the admin goes on the switch to execute some commands.

#interface vlan 1

#ip address 192.168.16.7 255.255.255.0

#no shutdown

#logging host 192.168.16.10

#exit

# Results

Basically we covered configuring standard and extended access control lists and the different ways to specify what computers to permit or deny. Then we configured ACLs on VTY lines to protect remote SSH or Telnet sessions. We also learned how to remove the applied ACLs as well as how to trouble shoot them if the necessary addresses aren’t being permitted or denied. I learned how to place ACLs because they do not always function properly if placed incorrectly. Afterwards, we enabled ftp and disabled icmp on the host with the use of access lists. The second part of the lab focused on dhcp pools and how to properly configure them on the router. We then were able to successfully verify the dhcp was working by going on the switch and dynamically assigning an address with the #ip address dhcp command. We worked through any conflicts that could possibly arise and we learned how to remove the binded ip address. Next we configured NAT, I never realized the importance of NAT and how crucial it is to match the clocks because this prevents the ping from providing false information about the metric and the hop count. Finally, there was the syslog (system log), and it needed to be configured to send messages to the system server.

# Application and Analysis

These concepts are to be applied in any field with any network size. I would change the redundancy of the commands in a live production environment. This lab was quite different from the previous labs because we were actually looking at the measures you would need to take to manage a network and keep it secure. Last, I think I have somewhat of an understanding between the concepts and the application; proof of this is visible in my procedure.

# Questions

1. ACLs and IP services were a new topic for me and it was interesting to see how network administrators have to go in and manually set access lists for each network. My discovery for this week’s lab was the wildcard mask. I thought it was for advertising networks in OSPF areas but I loved how ACLs used it to allow a group of users into the network or how you could allow a particular group and deny one person in that group from getting access to the network.

2. I didn’t have a chance to plan this lab; you can say that I planned for this lab by studying the material and participating in the discussing. Afterwards, it was just doing the lab and verifying that I had it done correctly. Albeit, it took me three attempts to fully understand everything.

3. The most difficult part of this lab was understanding the concepts, I remember there was a part of the lab where the order in which the ACLs were applied made the absolute difference in how they would accept or deny addresses, understanding these things took a lot of time and concentration.

4. Um everywhere!!

# Conclusion and Recommendation

This week we covered the foundation of network implementation. Before we were learning about things that you would do to create a network. Now these lessons are to actually bring the network to live production. These skills are useful in securing a network in a live production environment for any company that has network needs.

# Lessons Learned

I learned the importance of using ACLs and IP services; these two are key to implementing and securing a network. A less technical thing I learned is that using “?” after every phrase in a command is not only useful, but also necessary to do when you get to higher level commands because they are really easy to forget. This also prevents the admin from typing the incorrect commands.

# References

Class textbook