Access Lists

Introduction:

The technical name for an access list is Access Control List (ACL). The individual entries in an access control list are called access control entries. The term access control list isn't often used in practice; you'll typically hear these lists referred to simply as access lists or ACLs.

Access lists filter network traffic by controlling whether routed packets are forwarded or blocked at the router's interfaces. The router examines each packet to determine whether to forward or drop the packet, based on the criteria specified within the access lists.

Access list criteria could be the source address of the traffic, the destination address of the traffic, the upper-layer protocol, or other information.

There are many reasons to configure access list, for example, you can use access lists to restrict contents of routing updates, or to provide traffic flow control. But one of the most important reasons to configure access lists is to provide security for your network; this is the reason focused on in this experiment.

To create an access list, you specify the protocol to filter, you assign a unique name or number to the access list, and you define packet filtering criteria. A single access list can have multiple filtering criteria statements.

Named and numbered ACL:

Access lists on many Cisco devices can be either named or numbered. Named access lists are referenced with a name such as BZU. Numbered access lists are the older method, where each ACL is defined by a number such as 101.

Wildcard Masks:

Wildcard masks (also called inverse masks) are used in many devices for creating access lists. These masks can be confusing, because they're the opposite, in binary, of normal subnet masks. In other words, the wildcard

mask you would use to match a range that is described with a subnet mask of 255.255.255.0 would be 0.0.0.255.

Here's a simple rule that will solve most of the subnet/wildcard mask problems you'll see:

Replace all 0s with 255s, and all 255s with 0s.

The following table shows how class A, B, and C subnets masks are written as wild card masks:

Class Type	Subnet Mask	Wildcard Mask
A	255.0.0.0	0.255.255.255
B	255.255.0.0	0.0.255.255
C	255.255.255.0	0.0.0.255

Table 1: Classful wildcard masks

Standard ACL and Extended ACL:

Standard ACL use only the source IP address in an IP packet to filter the network. This basically permits or denies an entire suite of protocols. You can create a standard access list by using the number 1 to 99 .On the other hand extended ACL check for both source and destination IP address, protocol field in the Network layer header, and port number at the Transport layer header. You can use 100 to 199 for specifying your extended ACL.

The Implied "Deny All Traffic" Criteria Statement:

At the end of every access list is an implied "deny all traffic" criteria statement. Therefore, if a packet does not match any of your criteria statements, the packet will be blocked.

Configuring ACL:

This is the command syntax format of a standard ACL.

access-list access-list-number {permit|deny} {host|source source-wildcardmask|any}

For example, in your network you want that no computer or devices from 192.168.0.0/24 network can send traffic to your network. To implement this rule you need to write and ACL that will tell your router to discard all the traffic from 192.168.0.0/24. Now, let see how to implement this into a router using standard ACL

```
Router(config)#access-list 10 deny 192.168.0.0
0.0.0.255

Router(config)#access-list 10 permit any
```

After the ACL is defined whether it is standard or an extended one, it must be applied to the interface (inbound or outbound).

interface <interface>

ip access-group number {in|out}

```
Router(config)#interface fastEthernet 0/0

Router(config)#ip access-group 10 out
```

Unlike standard access control list, extended ACLs allow you to specify the source and destination IP address. Moreover, you can specify which protocols and service ports (i.e. www, telnet, and ftp) you want to deny in your router. For example you want to deny HTTP connection originating only from a host with IP 192.168.1.2 to your web server with IP 172.16.100.100, and to do that you have to write the following extended access control list on your router and then apply it to ai interface that you expect to receive incoming HTTP request from outsiders.

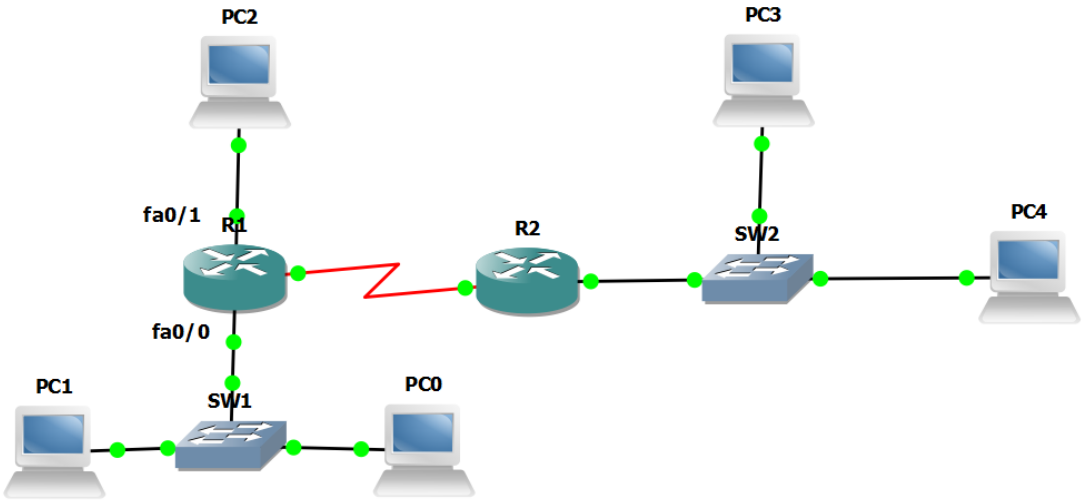
```
Router(config)# access-list 120 deny tcp host
192.168.1.2 host 172.16.100.100 eq 80

Router(config)# access-list 120 permit ip any any
```

If you did not add the “eq 80” in the above access list, then your router would deny all the tcp packets irrespective of its destination port, which means if a person try to FTP to your host he would be denied.

You can Issue the `show access-list` command in order to view the ACL entries.

Procedure:

Scenario: Build and configure the following topology based on the following requirements.



Requirements:

To help guide this initial configuration, you've assembled a list of requirements.

- Configure Routers R1 and R2 with the following configuration:

Interface	R1	R2
FastEthernet0/0 IP/SM	192.168.0.1/24	192.168.3.1/24
FastEthernet0/1 IP/SM	192.168.1.1/24	N/A
Serial IP/SM	192.168.2.1/30	192.168.2.2/30

- Configure PC0, PC1, PC2, PC3 and PC4 with the following configuration:

	PC0	PC1	PC2	PC3	PC4
Interface	NIC	NIC	NIC	NIC	NIC
IP/SM	192.168.0.2/24	192.168.0.3/24	192.168.1.2/24	192.168.3.2/24	192.168.3.3/24
Gateway	192.168.0.1	192.168.0.1	192.168.1.1	192.168.3.1	192.168.3.1

- Configure static routing or RIP on R1 and R2 and make sure all the PCs can reach each other.
- Enable Telnet on R1.

Tasks:

1) Use a standard ACL to deny PC2 from accessing the 192.168.3.0/24 network.

2) Use an extended ACL to deny PC3 **only** from remote telnet to R1.

3) The user on PC1 should be denied access to the serial link between R1 and R2; all other traffic should be allowed. Create and apply an access-list with no more than TWO statements to achieve this objective.