The University of Jordan (UJ)
School of Engineering
Department of Computer Engineering
Advanced Networks Lab 0907529
Exp.6 Access Control Lists (ACLs)

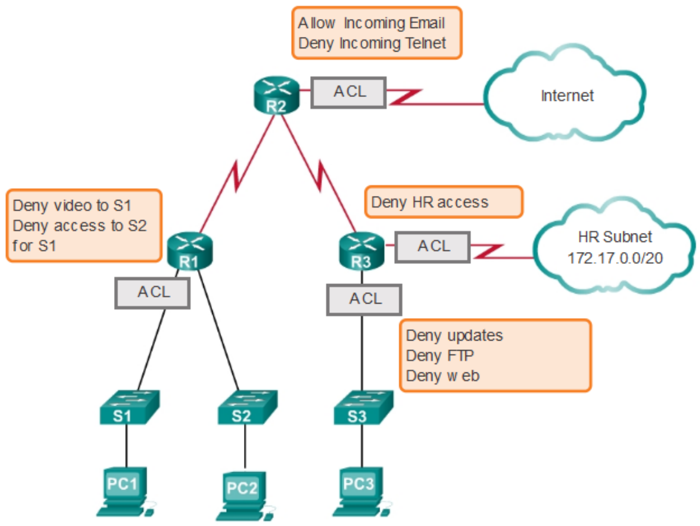
Objectives

1. Explain how ACLs are used to filter traffic.
2. Compare standard and extended IPv4 ACLs.
3. Explain the guidelines for creating and placement of ACLs.
4. Modify ACLs.
5. Explain how a router processes packets when an ACL is applied.
6. Troubleshoot common ACL errors.

1. Purpose of ACLs:

Network security is a huge subject, and one of the most important skills a network administrator needs is mastery of access control lists (ACLs).
Network designers use firewalls to protect networks from unauthorized use. Firewalls are hardware or software solutions that enforce network security policies. Consider a lock on a door to a room inside a building. The lock allows only authorized users with a key or access card to pass through the door. Similarly, a firewall filters unauthorized or potentially dangerous packets from entering the network. On a Cisco router, you can configure a simple firewall that provides basic traffic filtering capabilities using ACLs. Administrators use ACLs to stop traffic or permit only specified traffic on their networks.

An ACL is a sequential list of permit or deny statements that apply to addresses or upper-layer protocols based on information found in the packet header. ACLs provide a powerful way to control traffic into and out of a network. ACLs can be configured for all routed network protocols.
When configured, ACLs perform the following tasks as shown in the figure:



- Limit network traffic to increase network performance. For example, if corporate policy does not allow video traffic on the network, ACLs that block video traffic could be configured and applied. This would greatly reduce the network load and increase network performance.
- Provide traffic flow control. ACLs can restrict the delivery of routing updates. If updates are not required because of network conditions, bandwidth is preserved.
- Provide a basic level of security for network access. ACLs can allow one host to access a part of the network and prevent another host from accessing the same area. For example, access to the Human Resources network can be restricted to authorized users.
- Filter traffic based on traffic type. For example, an ACL can permit email traffic, but block all Telnet traffic.
- Screen hosts to permit or deny access to network services. ACLs can permit or deny a user to access file types, such as FTP or HTTP.

By default, a router does not have ACLs configured; therefore, by default a router does not filter traffic. Traffic that enters the router is routed solely based on information within the routing table. However, when an ACL is applied to an interface, the router performs the additional task of evaluating all network packets as they pass through the interface to determine if the packet can be forwarded.

An ACL is a sequential list of permit or deny statements, known as access control entries (ACEs). ACEs are also commonly called ACL statements. ACEs can be created to filter traffic based on certain criteria such as: the source address, destination address, the protocol, and port numbers. When network traffic passes through an interface configured with an ACL, the router compares the information within the packet against each ACE, in sequential order, to determine if the packet matches one of the statements. If a match is found, the packet is processed accordingly. In this way, ACLs can be configured to control access to a network or subnet.

To evaluate network traffic, the ACL extracts the following information from the Layer 3 packet header:

- Source IP address
- Destination IP address
- ICMP message type

The ACL can also extract upper layer information from the Layer 4 header, including:

- TCP/UDP source port
- TCP/UDP destination port

ACLs define the set of rules that give added control for packets that enter inbound interfaces, packets that relay through the router and packets that exit outbound interfaces of the router. ACLs do not act on packets that originate from the router itself.

ACLs are configured to apply to inbound traffic or to apply to outbound traffic as shown in the figure.

- Inbound ACLs - Incoming packets are processed before they are routed to the outbound interface. An inbound ACL is efficient because it saves the overhead of routing lookups if the packet is discarded. If the packet is permitted by the tests, it is then processed for routing. Inbound ACLs are best used to filter packets when the network attached to an inbound interface is the only source of the packets needed to be examined.
- Outbound ACLs - Incoming packets are routed to the outbound interface, and then they are processed through the outbound ACL. Outbound ACLs are best used when the same filter will be applied to packets coming from multiple inbound interfaces before exiting the same outbound interface.

The last statement of an ACL is always an implicit deny. This statement is automatically inserted at the end of each ACL even though it is not physically present. The implicit deny blocks all traffic. Because of this implicit deny, an ACL that does not have at least one permit statement will block all traffic.

2. <u>Standard versus Extended IPv4 ACLs:</u>

The two types of Cisco IPv4 ACLs are standard and extended. Standard ACLs can be used to permit or deny traffic only from source IPv4 addresses. The destination of the packet and the ports involved are not evaluated. The example below allows all traffic from the 192.168.30.0/24 network. Because of the implied "deny any" at the end, all other traffic is blocked with this ACL. Standard ACLs are created in global configuration mode.

R1(config)# access-list 10 permit 192.168.30.0 0.0.0.255

Extended ACLs filter IPv4 packets based on several attributes:

- Protocol type
- Source IPv4 address
- Destination IPv4 address
- Source TCP or UDP ports
- Destination TCP or UDP ports

In the example below, ACL 103 permits traffic originating from any address on the 192.168.30.0/24 network to any IPv4 network if the destination host port is 80 (HTTP). Extended ACLs are created in global configuration mode.
R1(config)# access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq80

The commands for standard and extended ACLs are explained in more details later in this experiment.

Standard and extended ACLs can be created using either a number or a name to identify the ACL and its list of statements. Using numbered ACLs is an effective method for determining the ACL type on smaller networks with more homogeneously defined traffic. However, a number does not provide information about the purpose of the ACL.

Numbered ACL: Assign a number based on protocol to be filtered.

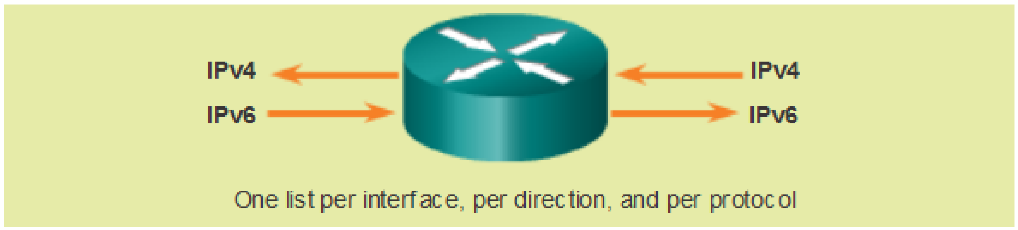
- (1 to 99) and (1300 to 1999): Standard IP ACL
- (100 to 199) and (2000 to 2699): Extended IP ACL

Named ACL: Assign a name to identify the ACL.

- Names can contain alphanumeric characters.
- It is suggested that the name be written in CAPITAL LETTERS.
- Names cannot contain spaces or punctuation.
- Entries can be added or deleted within the ACL.

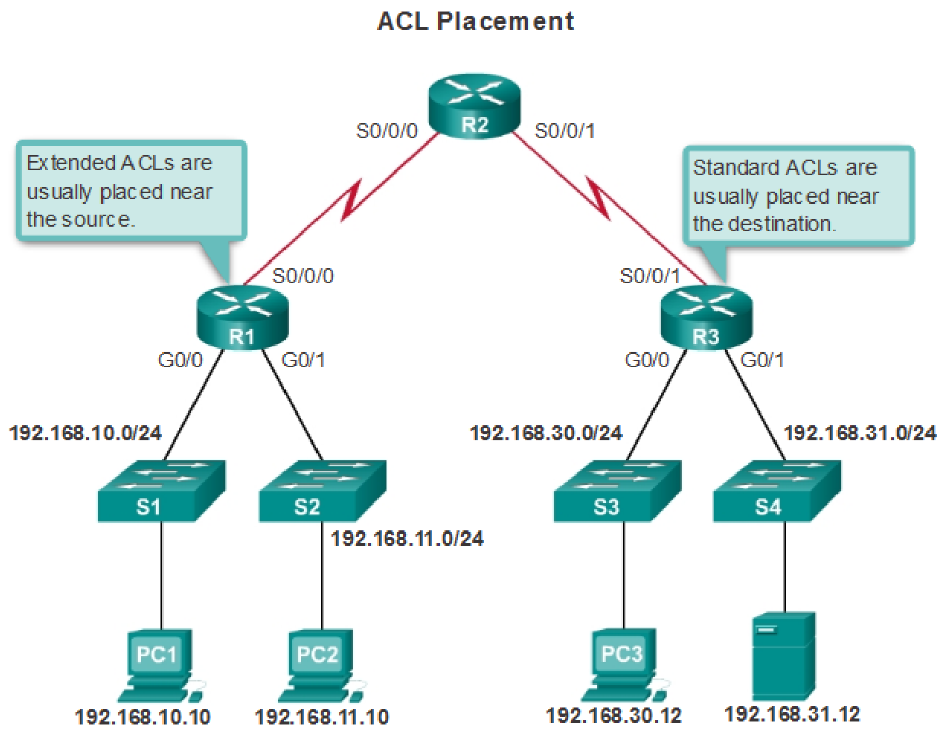
Writing ACLs can be a complex task. For every interface there may be multiple policies needed to manage the type of traffic allowed to enter or exit that interface. The router in the figure has two interfaces configured for IPv4 and IPv6. If we needed ACLs for both protocols, on both interfaces and in both directions, this would require eight separate ACLs. Each interface would have four ACLs; two ACLs for IPv4 and two ACLs for IPv6. For each protocol, one ACL is for inbound traffic and one for outbound traffic.
The Three Ps:
A general rule for applying ACLs on a router can be recalled by remembering the three Ps. You can configure one ACL per protocol, per direction, per interface:

- One ACL per protocol - To control traffic flow on an interface, an ACL must be defined for each protocol enabled on the interface.
- One ACL per direction - ACLs control traffic in one direction at a time on an interface. Two separate ACLs must be created to control inbound and outbound traffic.
- One ACL per interface - ACLs control traffic for an interface, for example, GigabitEthernet 0/0.



The proper placement of an ACL can make the network operate more efficiently. An ACL can be placed to reduce unnecessary traffic. For example, traffic that will be denied at a remote destination should not be forwarded using network resources along the route to that destination.
Every ACL should be placed where it has the greatest impact on efficiency. As shown in the figure below, the basic rules are:

- **Extended ACLs -** Locate extended ACLs as close as possible to the source of the traffic to be filtered. This way, undesirable traffic is denied close to the source network without crossing the network infrastructure.
- **Standard ACLs -** Because standard ACLs do not specify destination addresses, place them as close to the destination as possible. Placing a standard ACL at the source of the traffic will effectively prevent that traffic from reaching any other networks through the interface where the ACL is applied.

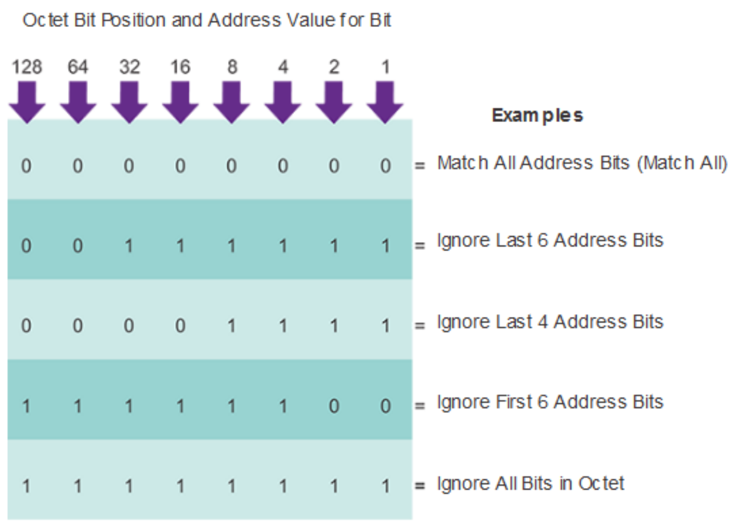


Wildcard Masks in ACLs
IPv4 ACEs include the use of wildcard masks. A wildcard mask is a string of 32 binary digits used by the router to determine which bits of the address to examine for a match.
Subnet masks use binary 1s and 0s to identify the network, subnet, and host portion of an IP address. Wildcard masks use binary 1s and 0s to filter individual IP addresses or groups of IP addresses to permit or deny access to resources.
Wildcard masks and subnet masks differ in the way they match binary 1s and 0s. Wildcard masks use the following rules to match binary 1s and 0s:

- Wildcard mask bit 0 - Match the corresponding bit value in the address.
- Wildcard mask bit 1 - Ignore the corresponding bit value in the address.

The following is an example on how to deal with wildcard masks:

	Decimal Address	Binary Address
IP Address to be processed	192.168.10.0	11000000.10101000.00001010.00000000
Wild Mask	0.0.255.255	00000000. 00000000.11111111. 11111111
Resulting IP Address	192.168.0.0	11000000.10101000. 00000000. 00000000


Octet Bit Position and Address Value for Bit
128 64 32 16 8 4 2 1
Examples
0 0 0 0 0 0 0 0 = Match All Address Bits (Match All)
0 0 1 1 1 1 1 1 = Ignore Last 6 Address Bits
0 0 0 0 1 1 1 1 = Ignore Last 4 Address Bits
1 1 1 1 1 1 0 0 = Ignore First 6 Address Bits
1 1 1 1 1 1 1 1 = Ignore All Bits in Octet

The ***any*** and *host* keywords: ***any*** keyword to substitute for the IPv4 address 0.0.0.0 with a wildcard mask of 255.255.255.255 as shown in the example below:
R1(config)# access-list 1 permit 0.0.0.0 255.255. 255.255
R1(config)# access-list 1 permit any
The ***host*** keyword to substitute for the wildcard mask when identifying a single host as shown in the example below.
R1(config)# access-list 1 permit 192.168.10.10 0.0.0.0
R1(config)# access-list 1 permit host 192.168.10.10

3. ACL Creation and Placement:

Configuring Standard ACLs:
To use numbered standard ACLs on a Cisco router, you must first create the standard ACL and then activate the ACL on an interface. The access-list global configuration command defines a standard ACL with a number in the range of 1 through 99. The full syntax of the standard ACL command is as follows:

Router(config)# **access-list** *access-list-number* { **deny** | **permit** } *source* [*source-wildcard*]

ACEs can deny or permit an individual host or a range of host addresses. If you want to remove the ACL, the global configuration **no access-list** command is used. Issuing the **show access-list** command confirms that access list 10 has been removed.

After a standard ACL is configured, it is linked to an interface using the **ip access-group** command in interface configuration mode:

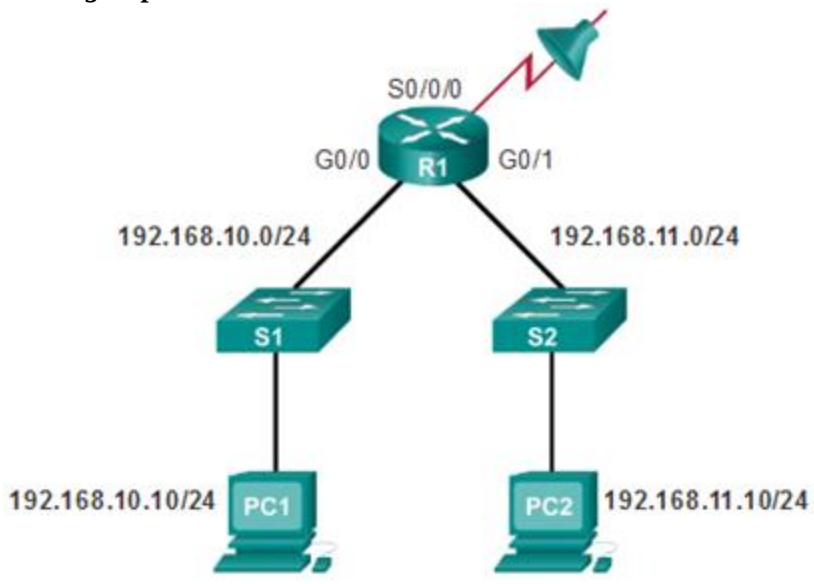
Router(config-if)# **ip access-group** { *access-list-number* | *access-list-name* } { **in** | **out** }

To remove an ACL from an interface, first enter the **no ip access-group** command on the interface, and then enter the global **no access-list** command to remove the entire ACL.

To create a statement that will permit a range of IPv4 addresses in a numbered ACL 10 that permits all IPv4 addresses in the network 192.168.10.0/24, you would enter:
R1(config)# access-list 10 permit 192.168.10.10 0.0.0.255

To apply a numbered standard ACL 10 on a router, you would enter:

R1(config)#interface serial 0/0/0
R1(config-if)# ip access-group 10 out



This ACL allows only traffic from source network 192.168.10.0 to be forwarded out of interface S0/0/0. Traffic from networks other than 192.168.10.0 is blocked.

Naming an ACL makes it easier to understand its function. For example, the above ACL could be called ***VLAN10_ALLOW***. When you identify your ACL with a name instead of with a number, the configuration mode and command syntax are slightly different as shown below.
R1(config)# ip access-list standard VLAN10_ALLOW
R1(config-std-nacl)# permit 192.168.10.10 0.0.0.255
R1(config-std-nacl)#exit
R1(config)#interface serial 0/0/0
R1(config-if)# ip access-group VLAN10_ALLOW out

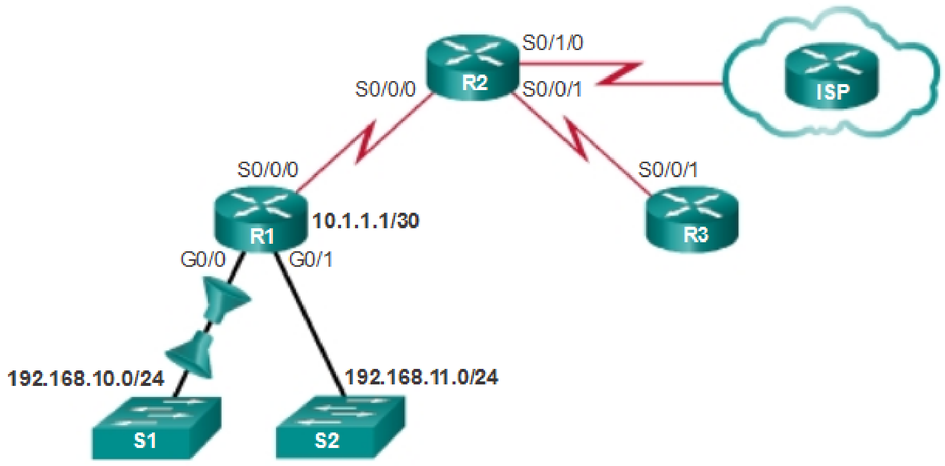
Using an ACL to Control VTY Access:
Restricting VTY access is a technique that allows you to define which IP addresses are allowed Telnet access to the router EXEC process. You can control which administrative workstation or network manages your router with an ACL and an **access-class** statement configured on your VTY lines.
An example allowing a range of addresses to access VTY lines 0 - 4 is shown below.
Configure the vty lines to accept incoming telnet connections using access list 21.
R1(config)# line vty 0 4
R1(config-line)# access-class 21 in
Create access list 21 to permit the 192.168.10.0/24 network to access VTY lines 0 - 4 and explicitly deny all others.
R1(config)# access-list 21 permit 192.168.10.0 0.0.0.255
R1(config)# access-list 21 deny any

Configuring Extended ACLs:
The procedural steps for configuring extended ACLs are the same as for standard ACLs. The extended ACL is first configured, and then it is activated on an interface. However, the command syntax and parameters are more complex to support the additional features provided by extended ACLs. The common command syntax for extended IPv4 ACLs is shown below. Note that there are many keywords and parameters for extended ACLs. It is not necessary to use all of the keywords and parameters when configuring an extended ACL.

Router(config)# ***access-list*** *access-list-number* { ***deny*** | ***permit*** } *protocol source* [*source-wildcard*] [***port*** *port-number or name*] *destination* [*destination-wildcard*] [***port*** *port-number or name*] [***established***]

The following is an example of an extended ACL. In this example, the network administrator has configured ACLs to restrict network access to allow website browsing only from the LAN attached to interface G0/0 to any external network. ACL 103 allows traffic coming from any address on the 192.168.10.0 network to go to any destination, subject to the limitation that the traffic is using ports 80 (HTTP) and 443 (HTTPS) only so that 192.168.10.0/24 network to browse both insecure and secure websites.

The nature of HTTP requires that traffic flow back into the network from websites accessed from internal clients. The network administrator wants to restrict that return traffic to HTTP exchanges from requested websites, while denying all other traffic. ACL 104 does that by blocking all incoming traffic, except for previously established connections. The permit statement in ACL 104 allows inbound traffic using the **established** parameter. Without the **established** parameter in the ACL statement, clients could send traffic to a web server, but not receive traffic returning from the web server.

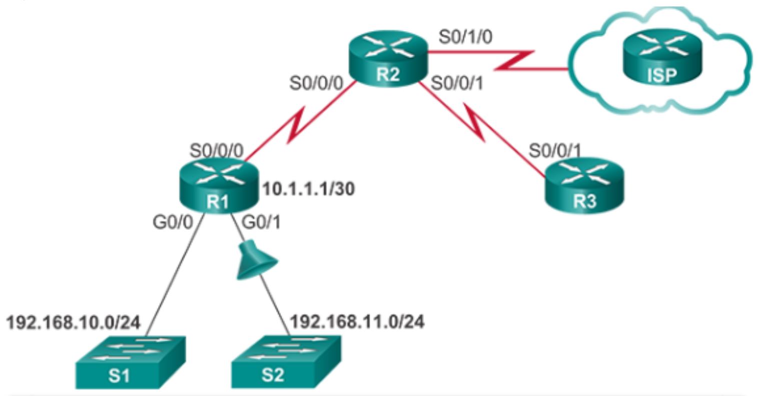


```
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)#access-list 104 permit tcp any 192.168.10.0 0.0.0.255 established
R1(config)#interface g0/0
R1(config-if)#ip access-group 103 in
R1(config-if)#ip access-group 104 out
```

- ACL 103 allows requests to ports 80 and 443.
- ACL 104 allows established HTTP and HTTPS replies.

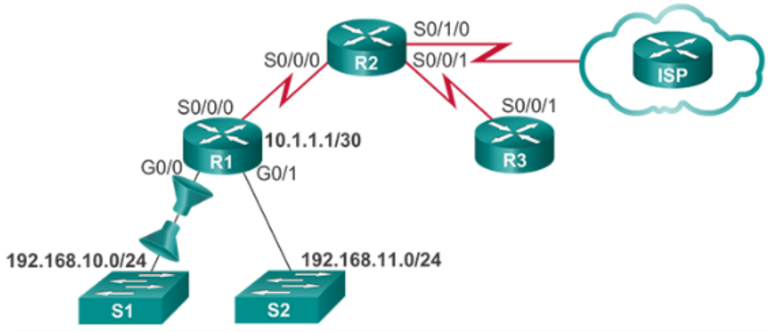
The example shown in the figure below denies FTP traffic from subnet 192.168.11.0 that is going to subnet 192.168.10.0, but permits all other traffic. Note the use of wildcard masks and the explicit deny any statement. Remember that FTP uses TCP ports 20 and 21; therefore the ACL requires both port name keywords **ftp** and **ftp-data** or **eq 20** and **eq 21** to deny FTP.
If using port numbers instead of port names, the commands would be written as:
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 20
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 21

To prevent the implied deny any statement at the end of the ACL from blocking all traffic, the **permit ip any any** statement is added. Without at least one **permit** statement in an ACL, all traffic on the interface where that ACL was applied would be dropped. The ACL should be applied inbound on the G0/1 interface so that traffic from the 192.168.11.0/24 LAN is filtered as it enters the router interface.



```
R1(config)# access-list 101 deny tcp 192.168.11.0 0.0.0.255
192.168.10.0 0.0.0.255 eq ftp
R1(config)# access-list 101 deny tcp 192.168.11.0 0.0.0.255
192.168.10.0 0.0.0.255 eq ftp-data
R1(config)# access-list 101 permit ip any any
R1(config)# interface g0/1
R1(config-if)#ip access-group 101 in
```

Named extended ACLs are created in essentially the same way that named standard ACLs are created. The figure below shows the named versions of the ACLs created in a previous example.



```
R1(config)#ip access-list extended SURFING
R1(config-ext-nacl)#permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config-ext-nacl)#permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config-ext-nacl)#exit
R1(config)#ip access-list extended BROWSING
R1(config-ext-nacl)#permit tcp any 192.168.10.0 0.0.0.255
established
R1(config-ext-nacl)#exit
R1(config)#interface g0/0
R1(config-if)#ip access-group SURFING in
R1(config-if)#ip access-group BROWSING out
```

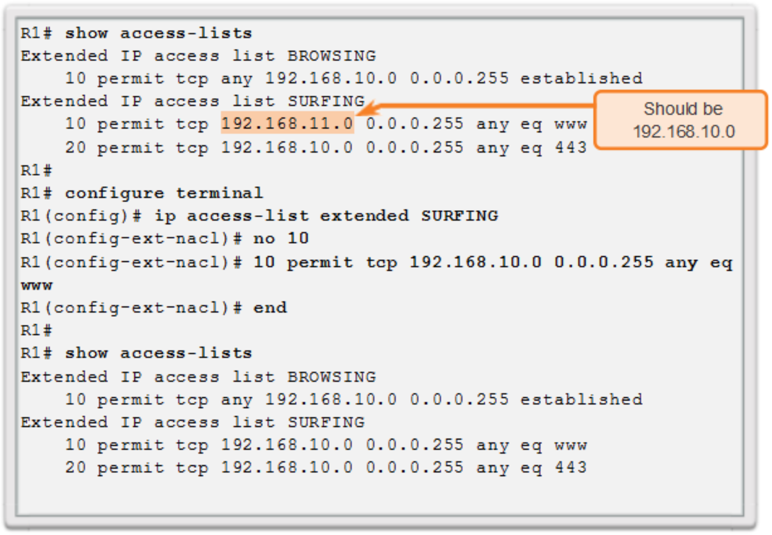
4. **Modify ACLs.**

Editing an extended ACL can be accomplished using the same process as editing a standard ACL. An ACL can be modified using:

- **Method 1 Text editor -** Using this method, the ACL is copied and pasted into the text editor where the changes are made. The current access list is removed using the **no access-list** command. The modified ACL is then pasted back into the configuration.

- **Method 2 Sequence numbers -** Sequence numbers can be used to delete or insert an ACL statement. The **ip access-list {standard | extended}** *name* command is used to enter named-ACL configuration mode. If the ACL is numbered instead of named, the ACL number is used in the *name* parameter. ACEs can be inserted or removed.

In the figure below the administrator needs to edit the ACL named SURFING to correct a typo in the source network statement. To view the current sequence numbers, the **show access-lists** command is used. The statement to be edited is identified as statement 10. The original statement is removed with the **no** *sequence_#* command. The corrected statement is added replacing the original statement.



```
R1# show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 192.168.11.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1#
R1# configure terminal
R1(config)# ip access-list extended SURFING
R1(config-ext-nacl)# no 10
R1(config-ext-nacl)# 10 permit tcp 192.168.10.0 0.0.0.255 any eq
www
R1(config-ext-nacl)# end
R1#
R1# show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 192.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
```

5. <u>Processing Packets with ACLs</u>

Inbound ACL Logic

If the information in a packet header and the first ACL statement match, the rest of the statements in the list are skipped, and the packet is permitted or denied as specified by the matched statement. If a packet header does not match an ACL statement, the packet is tested against the next statement in the list. This matching process continues until the end of the list is reached.

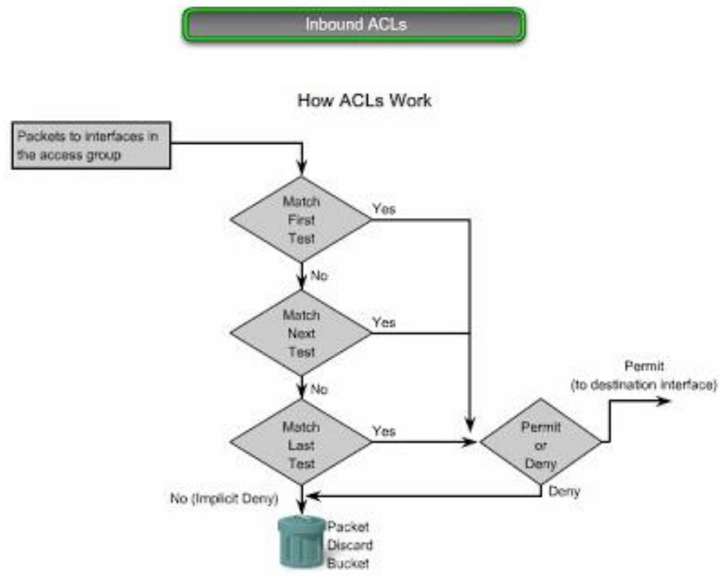
At the end of every ACL is a statement is an implicit *deny any* statement. This statement is not shown in output. This final implied statement applied to all packets for which conditions did not test true. This final test condition matches all other packets and results in a "deny" action. Instead of proceeding into or out of an interface, the router drops all of these remaining packets. This final statement is often referred to as the "implicit deny any statement" or the "deny all traffic" statement. Because of this statement, an ACL should have at least one permit statement in it; otherwise, the ACL blocks all traffic.

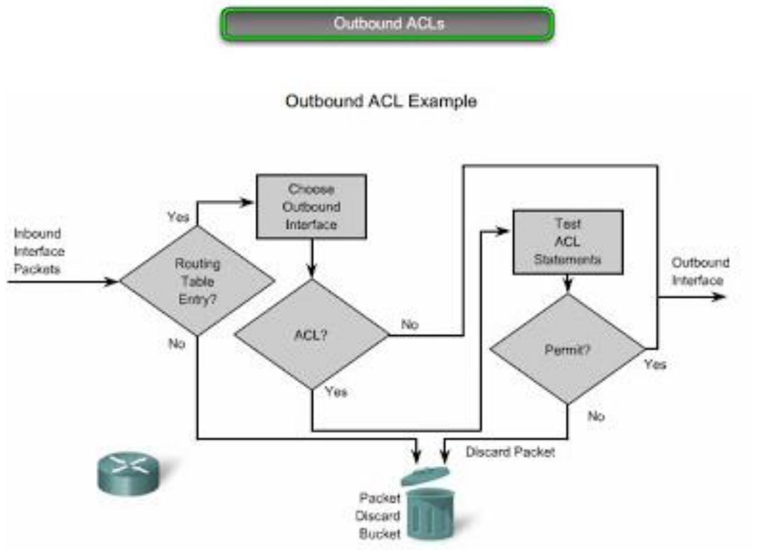
Outbound ACL Logic

Before a packet is forwarded to an outbound interface, the router checks the routing table to see if the packet is routable. If the packet is not routable, it is dropped and is not tested against the ACEs. Next, the router checks to see whether the outbound interface is grouped to an ACL. If the outbound interface is not grouped to an ACL, the packet can be sent to the output buffer. Examples of outbound ACL operation are as follows:

- **No ACL applied to the interface:** If the outbound interface is not grouped to an outbound ACL, the packet is sent directly to the outbound interface.
- **ACL applied to the interface:** If the outbound interface is grouped to an outbound ACL, the packet is not sent out on the outbound interface until it is tested by the combination of ACEs that are associated with that interface. Based on the ACL tests, the packet is permitted or denied.

For outbound lists, "permit" means to send the packet to the output buffer, and "deny" means to discard the packet.

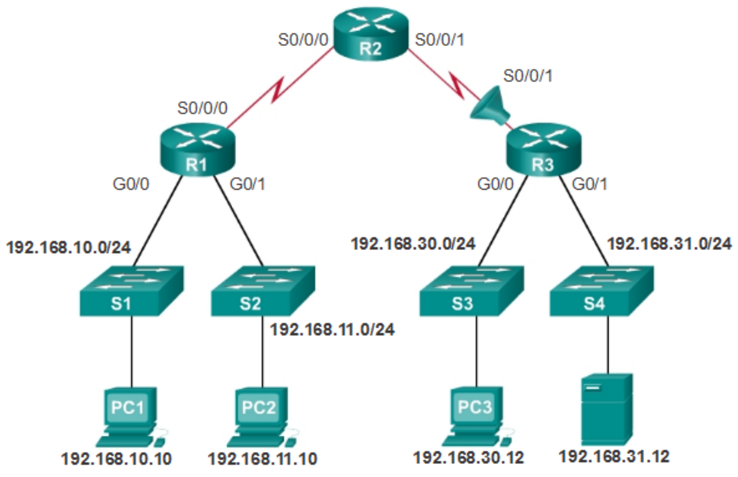


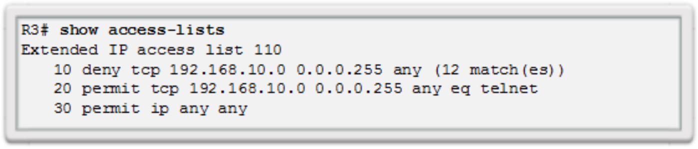


6. Troubleshoot ACLs

Using the **show** commands reveals most of the more common ACL errors. The most common errors are entering ACEs in the wrong order and not applying adequate criteria to the ACL rules.

Error Example 1
In the figure below, host 192.168.10.10 has no connectivity with 192.168.30.12. When viewing the output of the **show access-lists** command, matches are shown for the first deny statement. This is an indicator that this statement has been matched by traffic.
Solution - Look at the order of the ACEs (ACL Entries). Host 192.168.10.10 has no connectivity with 192.168.30.12 because of the order of rule 10 in the access list. Because the router processes ACLs from the top down, statement 10 denies host 192.168.10.10, so statement 20 can never be matched. Statements 10 and 20 should be reversed. The last line allows all other non-TCP traffic that falls under IP (ICMP, UDP, etc.).



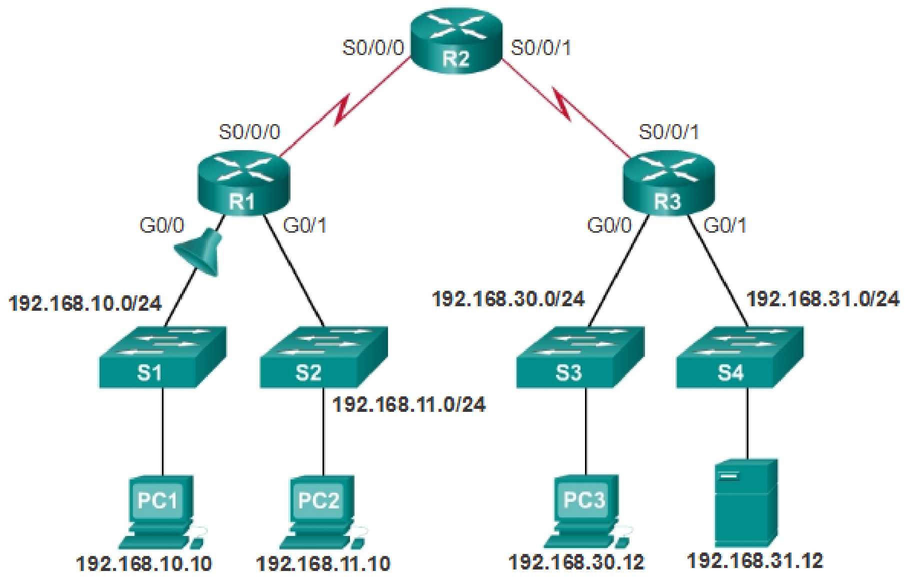

```
R3# show access-lists
Extended IP access list 110
    10 deny tcp 192.168.10.0 0.0.0.255 any (12 match(es))
    20 permit tcp 192.168.10.0 0.0.0.255 any eq telnet
    30 permit ip any any
```

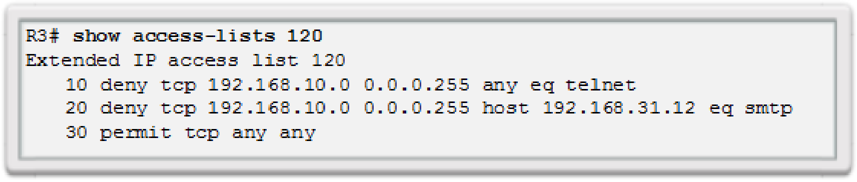
Error Example 2

In the figure above, the 192.168.10.0/24 network cannot use TFTP to connect to the 192.168.30.0/24 network.

Solution - The 192.168.10.0/24 network cannot use TFTP to connect to the 192.168.30.0/24 network because TFTP uses the transport protocol UDP. Statement 30 in access list 120 allows all other TCP traffic. However, because TFTP uses UDP instead of TCP, it is implicitly denied. Recall that the implied deny any statement does not appear in **show access-lists** output and therefore matches are not shown. Statement 30 should be **ip any any**.

This ACL works whether it is applied to G0/0 of R1, or S0/0/1 of R3, or S0/0/0 of R2 in the incoming direction. However, based on the rule about placing extended ACLs closest to the source, the best option is to place it inbound on G0/0 of R1 because it allows undesirable traffic to be filtered without crossing the network infrastructure.



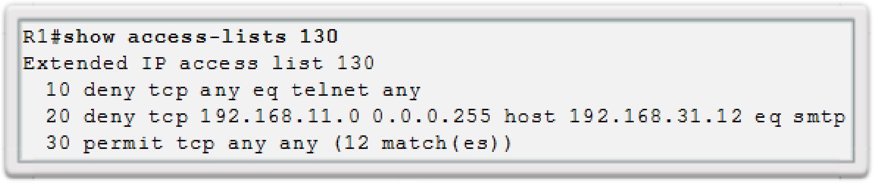


```
R3# show access-lists 120
Extended IP access list 120
    10 deny tcp 192.168.10.0 0.0.0.255 any eq telnet
    20 deny tcp 192.168.10.0 0.0.0.255 host 192.168.31.12 eq smtp
    30 permit tcp any any
```

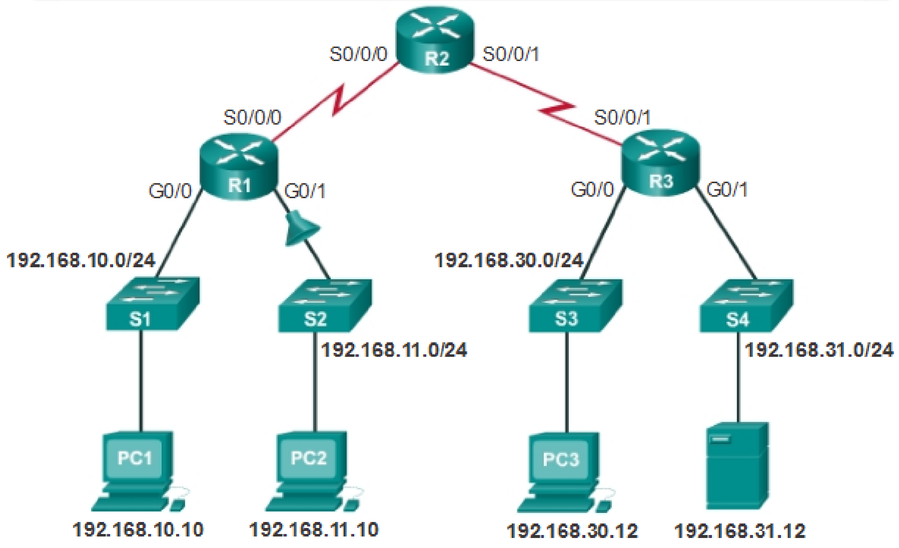
Error Example 3

In the figure above, the 192.168.11.0/24 network can use Telnet to connect to 192.168.30.0/24, but according to company policy, this connection should not be allowed. The results of the **show access-lists 130** command indicate that the permit statement has been matched.

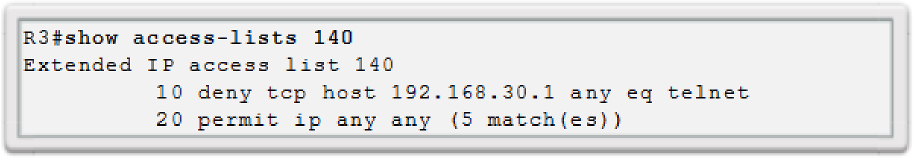
Solution - The 192.168.11.0/24 network can use Telnet to connect to the 192.168.30.0/24 network, because the Telnet port number in statement 10 of access list 130 is listed in the wrong position in the ACL statement. Statement 10 currently denies any source packet with a port number that is equal to Telnet. To deny Telnet traffic inbound on G0/1, deny the destination port number that is equal to Telnet, for example, **deny tcp any any eq telnet**.



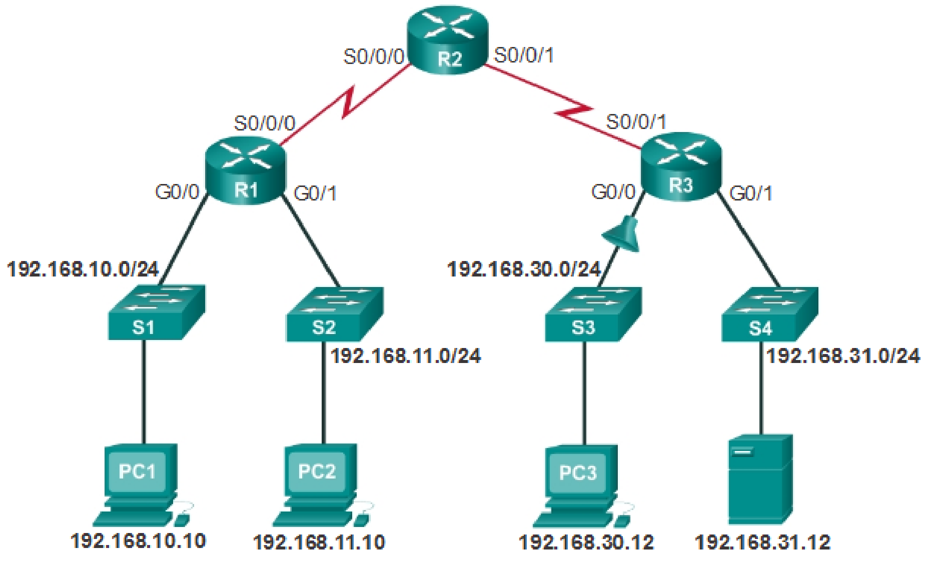
```
R1#show access-lists 130
Extended IP access list 130
  10 deny tcp any eq telnet any
  20 deny tcp 192.168.11.0 0.0.0.255 host 192.168.31.12 eq smtp
  30 permit tcp any any (12 match(es))
```



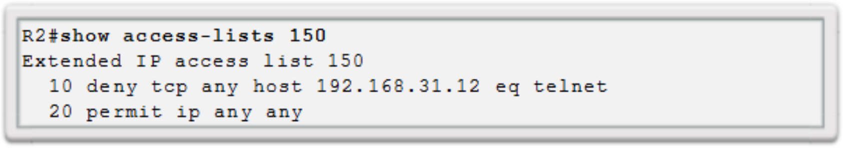
Error Example 4
In the figure below, host 192.168.30.12 is able to Telnet to connect to 192.168.31.12, but company policy states that this connection should not be allowed. Output from the **show access-lists 140** command indicate that the permit statement has been matched.
Solution - Host 192.168.30.12 can use Telnet to connect to 192.168.31.12 because there are no rules that deny host 192.168.30.12 or its network as the source. Statement 10 of access list 140 denies the router interface on which traffic enters the router. The host IPv4 address in statement 10 should be 192.168.30.12.



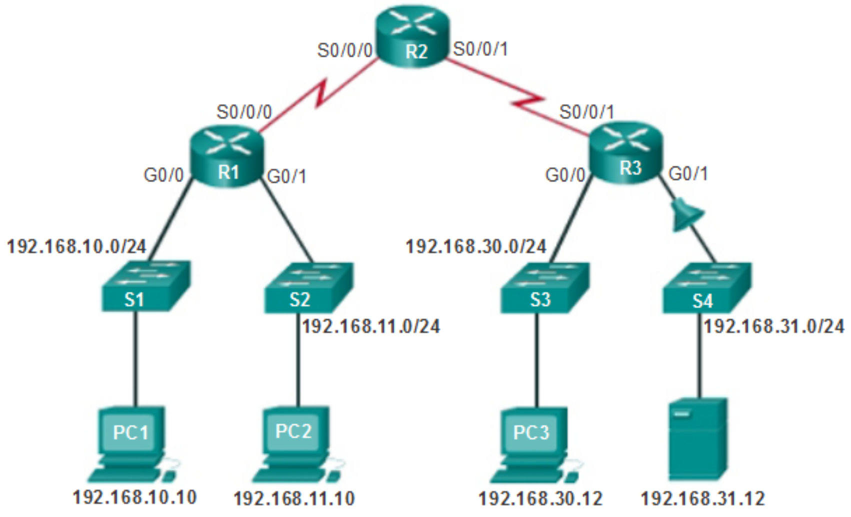
```
R3#show access-lists 140
Extended IP access list 140
          10 deny tcp host 192.168.30.1 any eq telnet
          20 permit ip any any (5 match(es))
```



Error Example 5
In the figure below, host 192.168.30.12 can use Telnet to connect to 192.168.31.12, but according to the security policy, this connection should not be allowed. Output from the **show access-lists 150** command indicate that no matches have occurred for the deny statement as expected.
Solution - Host 192.168.30.12 can use Telnet to connect to 192.168.31.12 because of the direction in which access list 150 is applied to the G0/1 interface. Statement 10 denies any source address to connect to host 192.168.31.12 using telnet. However, this filter should be applied outbound on G0/1 to filter correctly.



```
R2#show access-lists 150
Extended IP access list 150
  10 deny tcp any host 192.168.31.12 eq telnet
  20 permit ip any any
```



Procedures:
You can find the lab problems sheet, the packet tracer activities and the practical discussion videos on your Microsoft Teams group.

Reference:
Enterprise Networking, Security, and Automation - Cisco Networking Academy