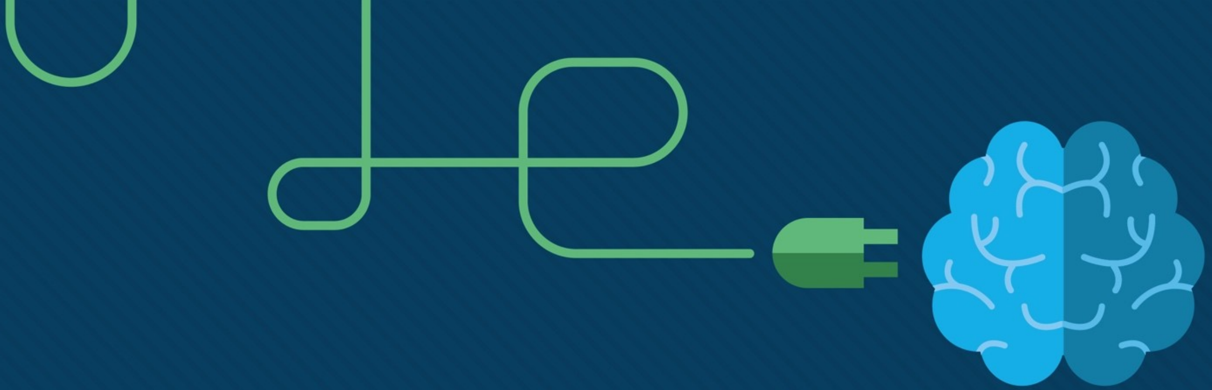


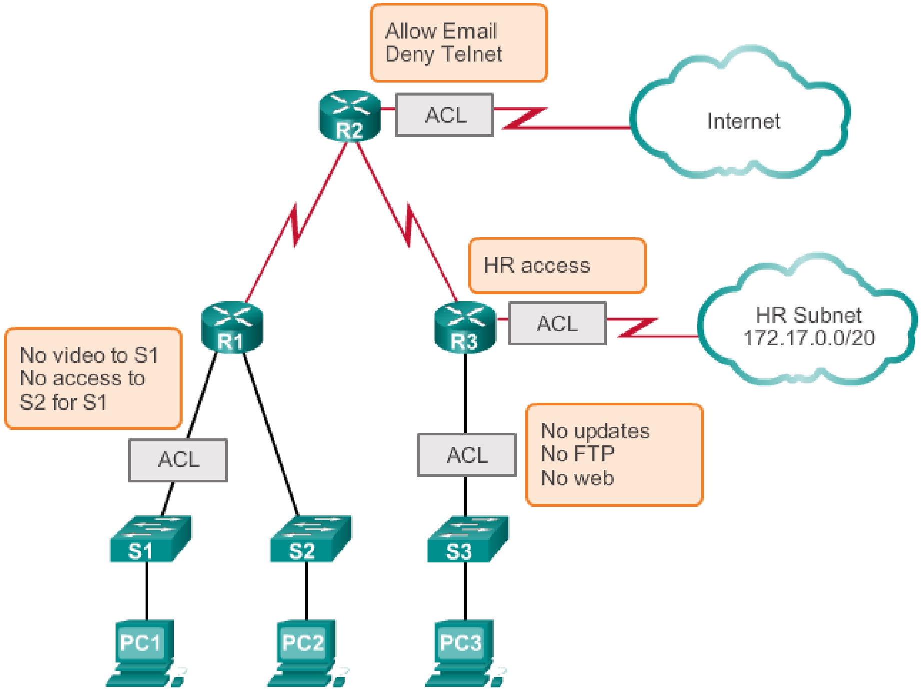
Access Control Lists

CCNA Routing and Switching

Connecting Networks v6.0

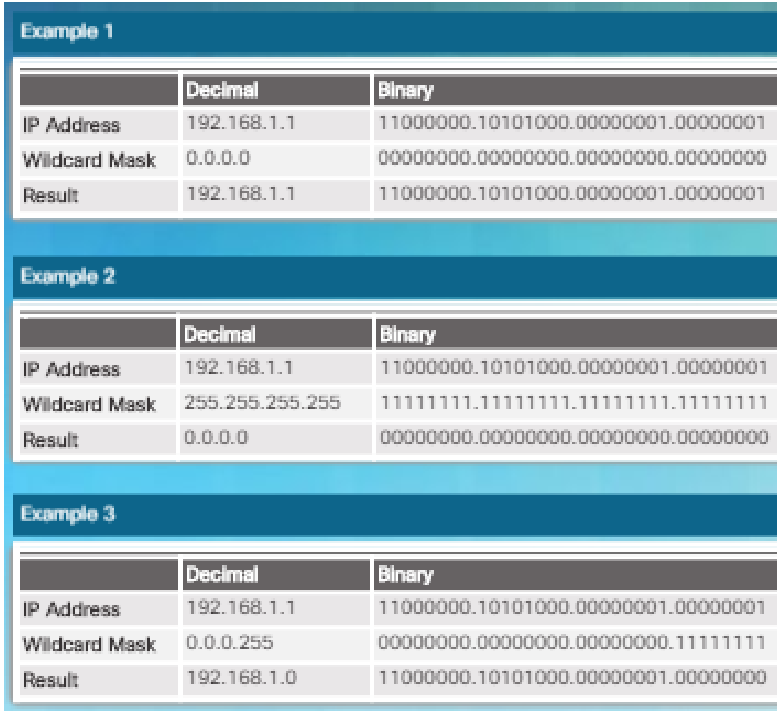
Access Control Lists (ACLs)

- By default,
  - a router does not filter traffic.
- When an ACL is applied to an interface, it
  - evaluates all network packets
  - determines if the packet is permitted or denied.



ACL Operation Overview

- An ACL contains a sequential list of **permit** or **deny** statements, known as access control entries (ACEs).
  - ACEs are also commonly called *ACL statements*.
  - IPv4 ACEs include the use of wildcard masks which are a string of 32 binary digits used by the router to determine which bits of the address to examine for a match.



Example 1

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.0	00000000.00000000.00000000.00000000
Result	192.168.1.1	11000000.10101000.00000001.00000001

Example 2

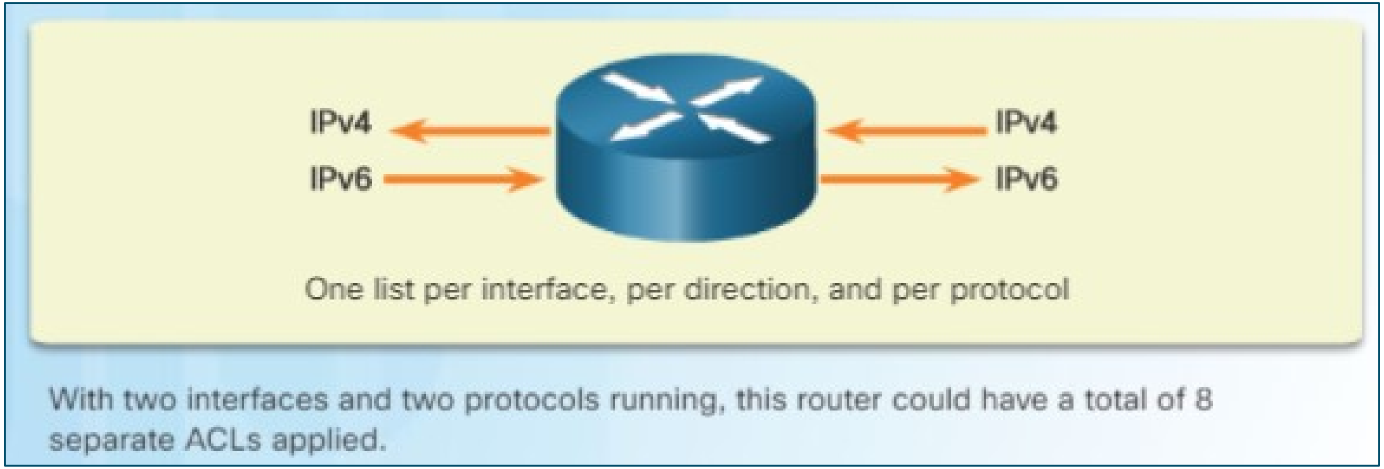
	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	255.255.255.255	11111111.11111111.11111111.11111111
Result	0.0.0.0	00000000.00000000.00000000.00000000

Example 3

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.255	00000000.00000000.00000000.11111111
Result	192.168.1.0	11000000.10101000.00000001.00000000

ACL Operation Overview

- You can configure:
  - **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.

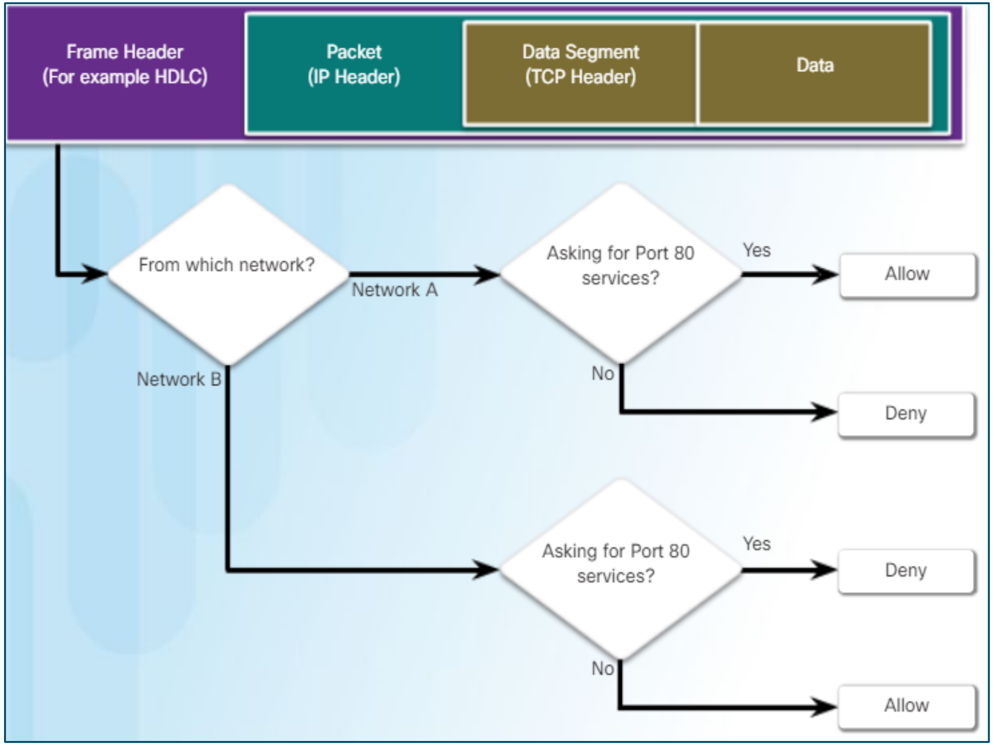


One list per interface, per direction, and per protocol

With two interfaces and two protocols running, this router could have a total of 8 separate ACLs applied.

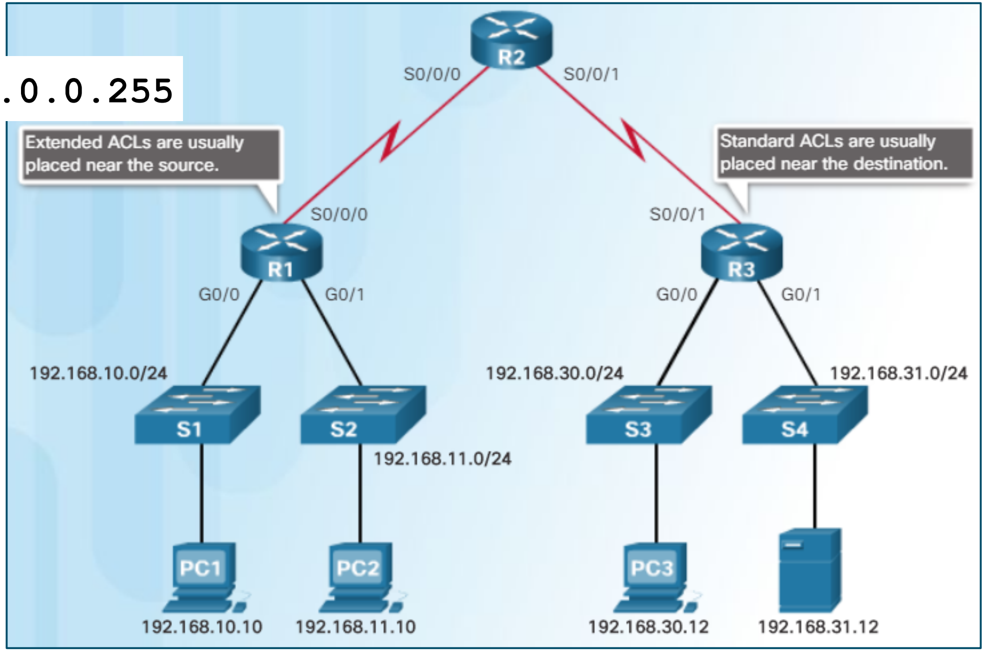
ACL Operation Overview

- To help explain how an ACL operates, refer to the decision path used to filter web traffic.
- An ACL has been configured to:
  - Permit web access to users from Network A but deny all other services to Network A users.
  - Deny HTTP access to users from Network B, but permit network B users to have all other access.



Types of IPv4 ACLs

- **Standard ACLs** filter packets based on:
  - the source address only.
  - **`access-list 10 permit 192.168.30.0 0.0.0.255`**
  - located as close to the destination as possible.



Extended ACLs are usually placed near the source.

Standard ACLs are usually placed near the destination.

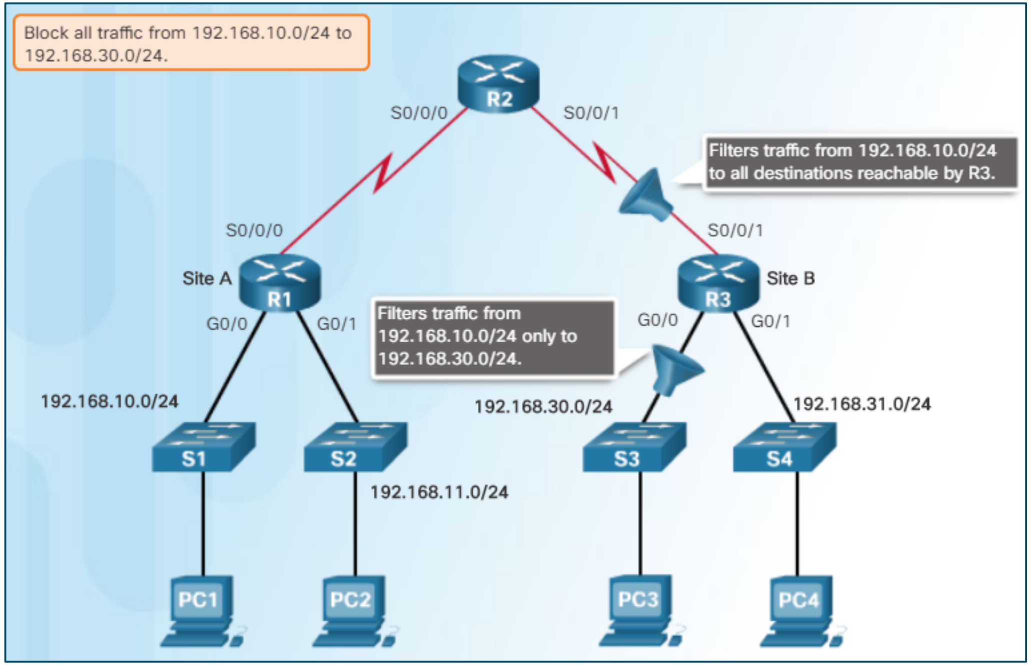
- **Extended ACLs** filter packets based on:
  - Protocol type / Protocol number (e.g., IP, ICMP, UDP, TCP, …)
  - Source and destination IP addresses
  - Source and Destination TCP and UDP ports
  - **`access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq 80`**
  - located as close as possible to the source of the traffic to be filtered.

Types of IPv4 ACLs

- Standard and extended ACLs can be created using either a number or a name to identify the ACL and its list of statements.
- 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.

Standard ACL

- A standard ACL will be configured to block all traffic from 192.168.10.0/24 going to 192.168.30.0/24.
- The standard ACL should be applied closest to the destination and therefore could be applied outgoing on the R3 G0/0 interface.
  - Applying it incoming on the R3 S0/0/1 interface would prevent reaching 192.168.31.0/24 and therefore should not be applied to this interface.
- If a standard ACL was placed at the source of the traffic, it would filter traffic based on the given source address no matter where the traffic is destined.



Block all traffic from 192.168.10.0/24 to 192.168.30.0/24.

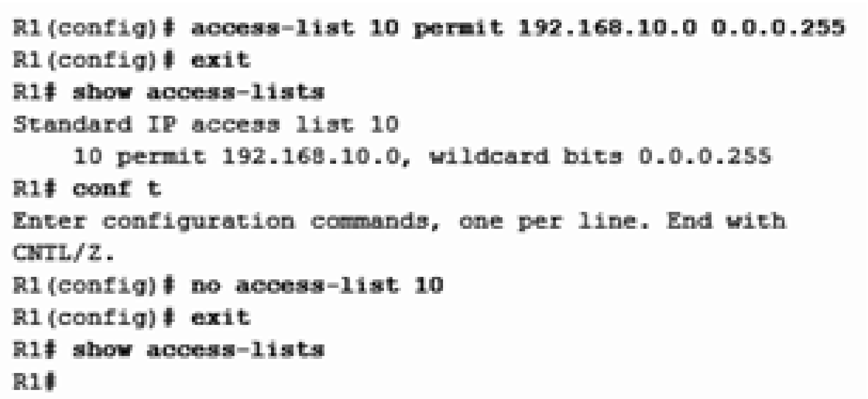
Filters traffic from 192.168.10.0/24 to all destinations reachable by R3.

Filters traffic from 192.168.10.0/24 only to 192.168.30.0/24.

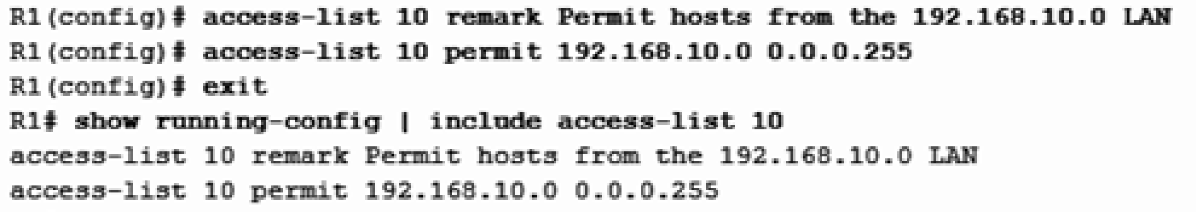
Standard IPv4 ACL Implementation

- The full syntax of the standard ACL command is as follows:
  - **access-list** *ACL-#* {**deny** | **permit** | **remark**} *source* [*source-wildcard*][**log**]

- For example:
  - Permit all IP addresses in network 192.168.10.0/24
  - Use the **no access-list 10** command to remove an ACL.
  - Use the **remark** keyword for documenting an ACL to make it easier to understand.



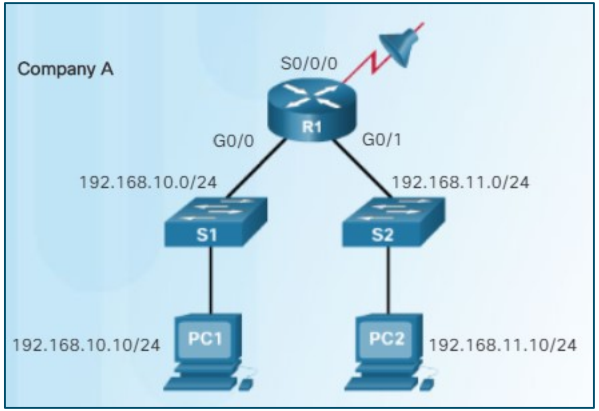
```
R1(config)# access-list 10 permit 192.168.10.0 0.0.0.255
R1(config)# exit
R1# show access-lists
Standard IP access list 10
    10 permit 192.168.10.0, wildcard bits 0.0.0.255
R1# conf t
Enter configuration commands, one per line. End with
CNTL/Z.
R1(config)# no access-list 10
R1(config)# exit
R1# show access-lists
R1#
```

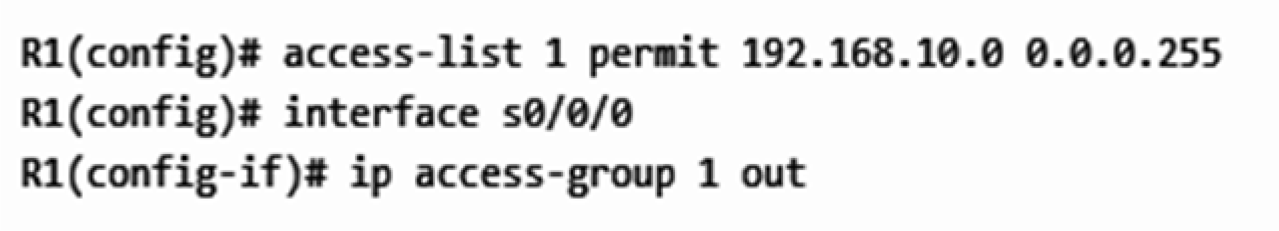


```
R1(config)# access-list 10 remark Permit hosts from the 192.168.10.0 LAN
R1(config)# access-list 10 permit 192.168.10.0 0.0.0.255
R1(config)# exit
R1# show running-config | include access-list 10
access-list 10 remark Permit hosts from the 192.168.10.0 LAN
access-list 10 permit 192.168.10.0 0.0.0.255
```

Standard IPv4 ACL Implementation

- An IPv4 ACL is linked to an interface using the following interface configuration mode command:
  - **ip access-group** {*ACL-#* | *access-list-name*} {**in** | **out**}



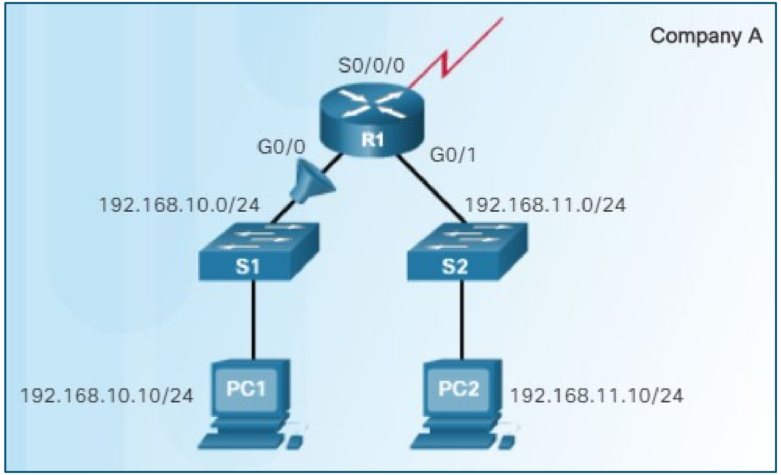


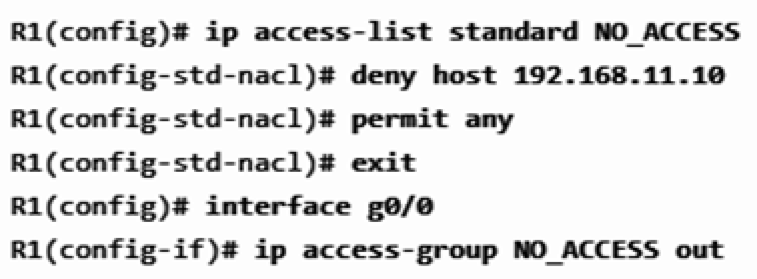
```
R1(config)# access-list 1 permit 192.168.10.0 0.0.0.255
R1(config)# interface s0/0/0
R1(config-if)# ip access-group 1 out
```

- **Note**:
  - 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.

Standard IPv4 ACL Implementation

- To create a standard **named** ACL.
  - Use the **`ip access-list standard`** *`name`* global config command.
    - Names are alphanumeric, case sensitive, and must be unique.
    - The command enters standard named ACL configuration mode.
  - Use **`permit`**, **`deny`**, or **`remark`** statements.
  - Apply the ACL to an interface using the **`ip access-group`** *`name`* command.

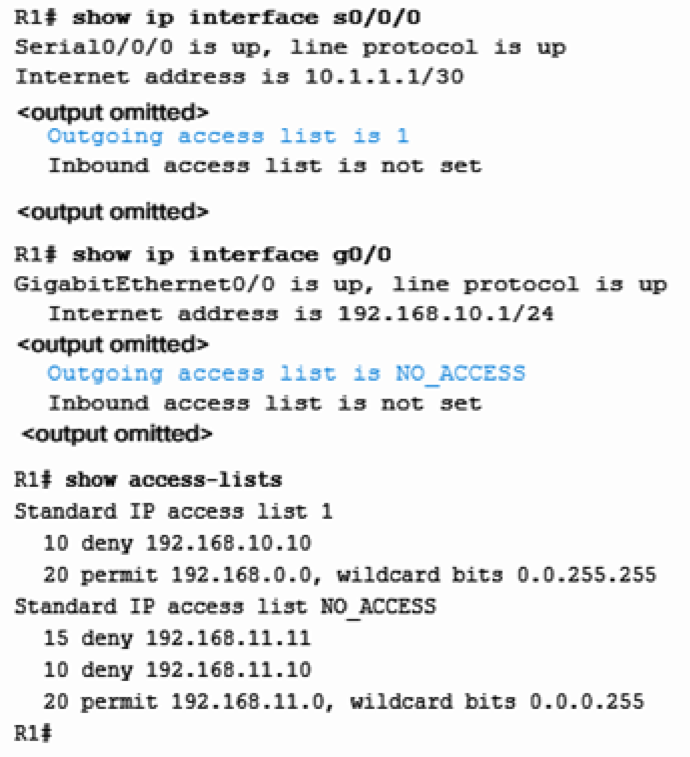




```
R1(config)# ip access-list standard NO_ACCESS
R1(config-std-nacl)# deny host 192.168.11.10
R1(config-std-nacl)# permit any
R1(config-std-nacl)# exit
R1(config)# interface g0/0
R1(config-if)# ip access-group NO_ACCESS out
```

Standard IPv4 ACL Implementation

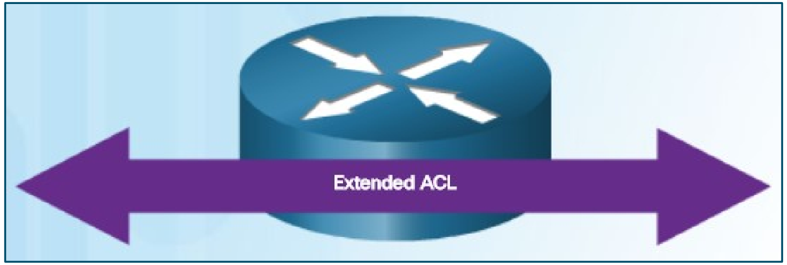
- Use the **show ip interface** command to verify the ACL on the interface.
  - The output includes the number or name of the access list and the direction in which the ACL was applied.
- Use the **show access-lists** [*ACL-#* | *access-list-name*] command to view the content of a standard ACL.
  - Notice that the NO_ACCESS statements are out of order because Cisco IOS uses a special hashing function for standard ACLs and re-orders host ACEs so they are processed first optimizing the search for a host ACL entry.
  - Standard ACLs process network ACEs in the order in which they were entered.



```
R1# show ip interface s0/0/0
Serial0/0/0 is up, line protocol is up
Internet address is 10.1.1.1/30
<output omitted>
  Outgoing access list is 1
  Inbound access list is not set

<output omitted>

R1# show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet address is 192.168.10.1/24
<output omitted>
  Outgoing access list is NO_ACCESS
  Inbound access list is not set
 <output omitted>

R1# show access-lists
Standard IP access list 1
  10 deny 192.168.10.10
  20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO_ACCESS
  15 deny 192.168.11.11
  10 deny 192.168.11.10
  20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

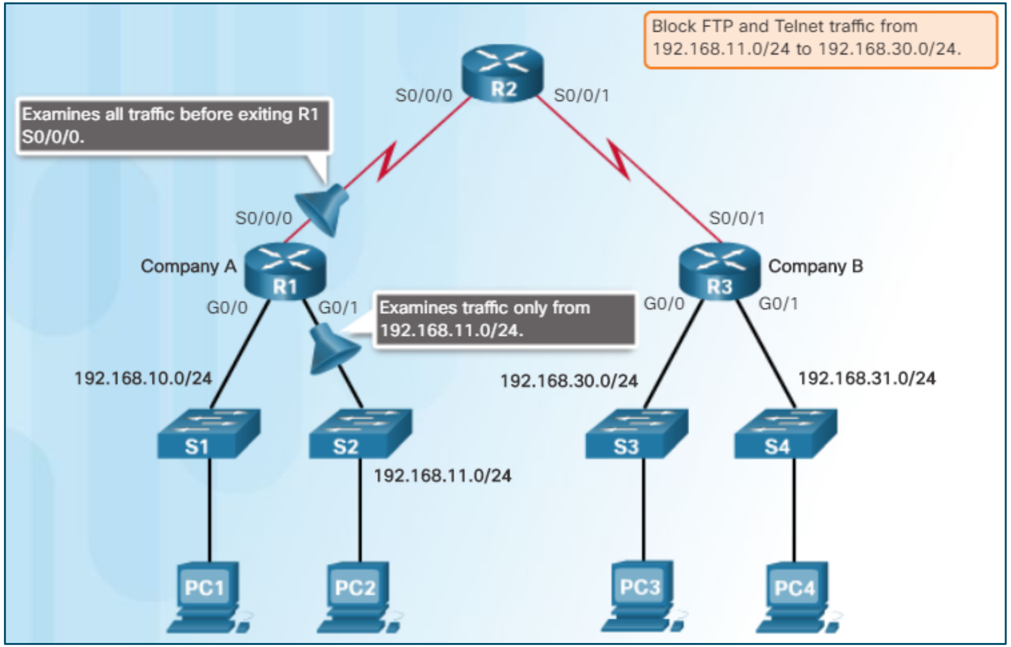
Extended ACL

- Extended IPv4 ACLs provide more precise filtering.
  - Extended ACLs are numbered 100 to 199 and 2000 to 2699, providing a total of 799 possible extended numbered ACLs.
  - Extended ACLs can also be named.
  - Extended ACLs are used more often than standard ACLs because they provide a greater degree of control.
- Extended ACLs can filter on:
  - Source address
  - Destination address
  - Protocol
  - Port number



Extended ACL

- An extended ACL will be configured to block all FTP and Telnet traffic from 192.168.11.0/24 going to 192.168.30.0/24.
- The extended ACL should be applied closest to the source and therefore could be applied incoming on the R1 G0/1 interface.
  - Applying it outgoing on the R1 S0/0/1 interface would prevent reaching 192.168.31.0/24 but would also needlessly process packets from 192.168.10.0/24.
- If a extended ACL is placed at the source of the traffic, it would deny undesirable traffic close to the source network without crossing the network infrastructure.



Block FTP and Telnet traffic from 192.168.11.0/24 to 192.168.30.0/24.

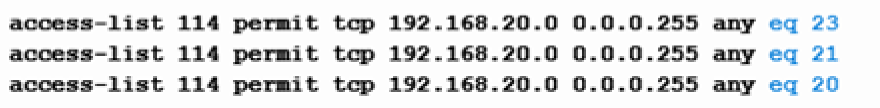
Examines all traffic before exiting R1 S0/0/0.

Examines traffic only from 192.168.11.0/24.

Extended IPv4 ACLs

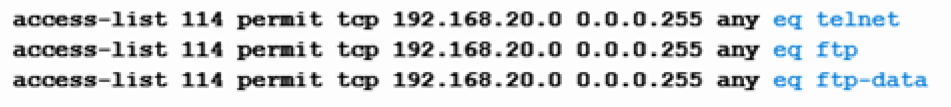
Structure of an Extended IPv4 ACLs

- Extended ACLs can filter on protocol and port number.
- An application can be specified by configuring either:
  - The port number



```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 23
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 21
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 20
```

  - The name of a well-known port.

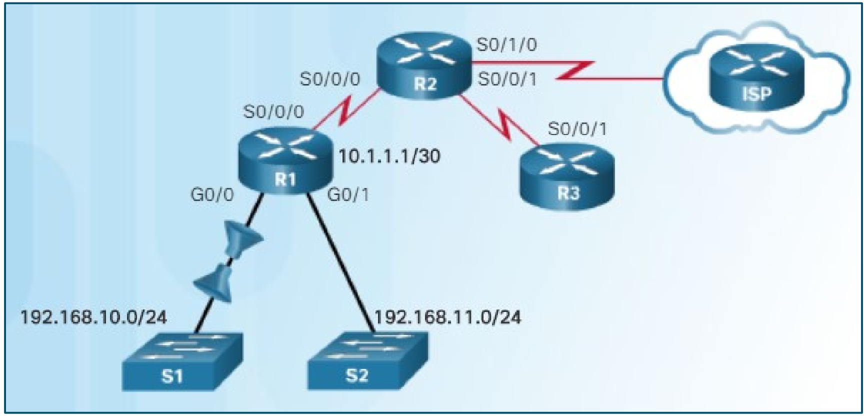


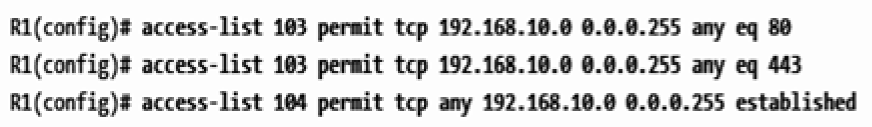
```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq telnet
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp-data
```

- Note:
  - Use the question mark (?) to see available well-known port names.
  - E.g., **`access-list 101 permit tcp any any eq ?`**

Configure Extended IPv4 ACLs

- The full syntax of the extended ACL command is as follows:
  - **access-list** *ACL-#* {**deny** | **permit** | **remark**} *protocol* {*source source-wildcard*][*operator* [*port-number* | *port-name*]] {*destination destination-wildcard*][*operator* [*port-number* | *port-name*]]
- For example:
  - ACL 103 allows requests to port 80 and 443.
  - ACL 104 allows established HTTP and HTTPS replies.
  - The **established** parameter allows only responses to traffic that originates from the 192.168.10.0/24 network to return to that network.

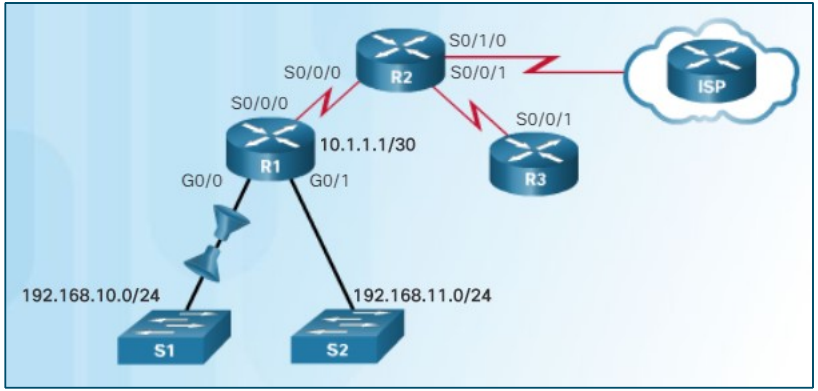


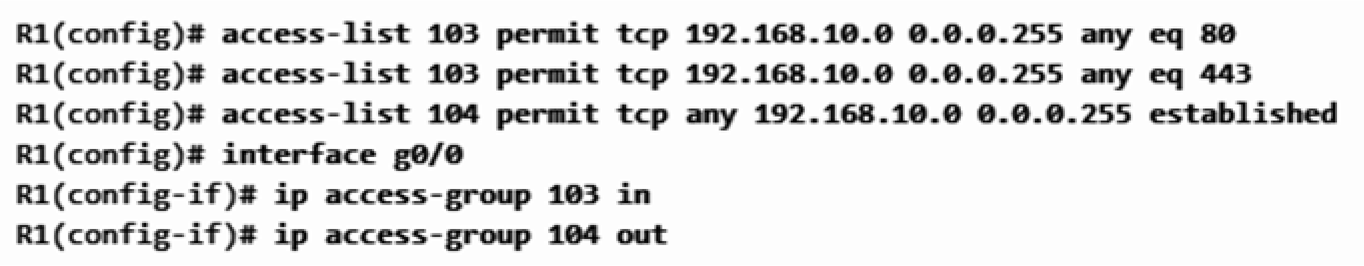


```
R1(config)# access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)# access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)# access-list 104 permit tcp any 192.168.10.0 0.0.0.255 established
```

Configure Extended IPv4 ACLs

- Applying extended ACLs is similar to standard ACLs except that they should be applied as close to the source.

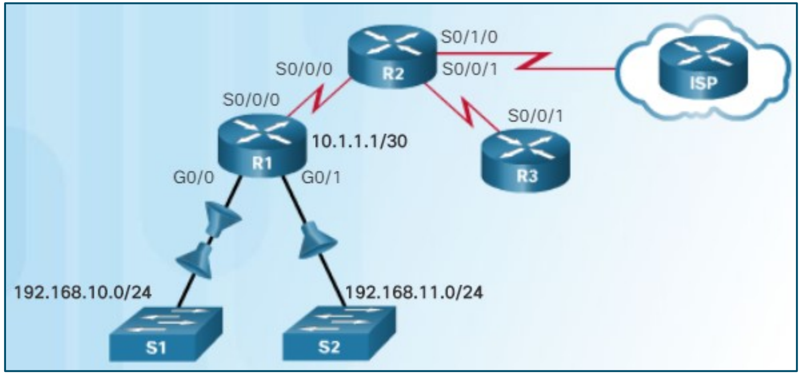


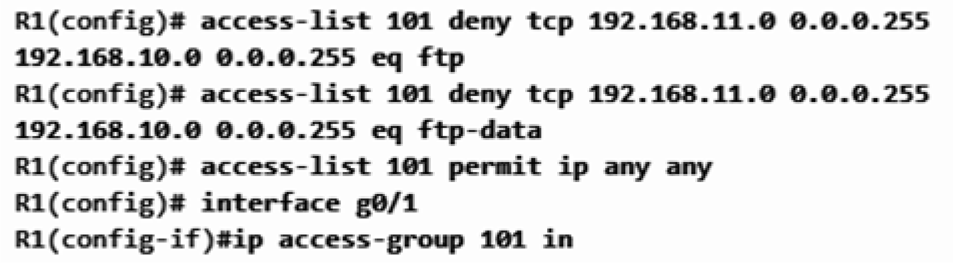


```
R1(config)# access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)# access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)# access-list 104 permit tcp any 192.168.10.0 0.0.0.255 established
R1(config)# interface g0/0
R1(config-if)# ip access-group 103 in
R1(config-if)#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

Configure Extended IPv4 ACLs

- In this example, FTP traffic from subnet 192.168.11.0 going to subnet 192.168.10.0 is denied, but all other traffic is permitted.
  - FTP utilizes two port numbers (TCP port 20 and 21) therefore two ACEs are required.
  - The example uses the well-known port names **ftp** and **ftp-data**.
  - Without at least one permit statement in an ACL, all traffic on the interface where that ACL was applied would be dropped.
  - The ACL is applied incoming on the R1 G0/1 interface.

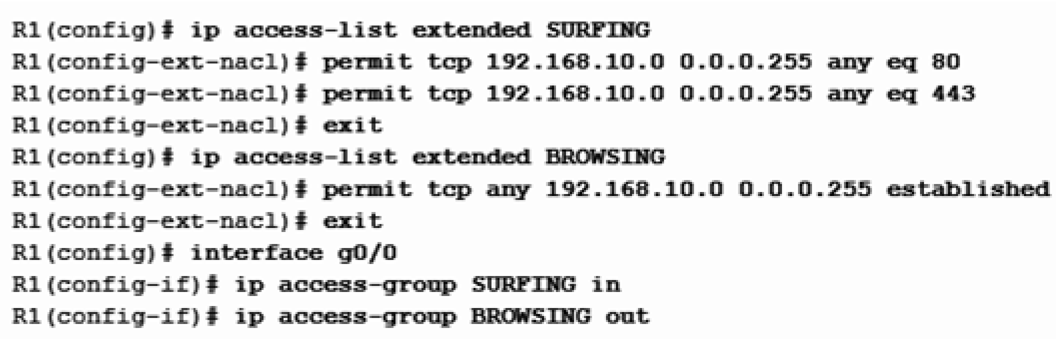




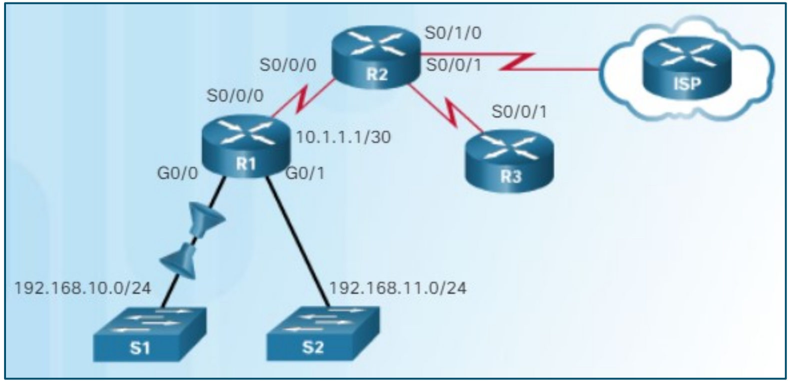
```
R1(config)# access-list 101 deny tcp 192.168.11.0 0.0.0.255
192.168.10.0 0.0.0.255 eq ftp
R1(config)# access-list 101 deny tcp 192.168.11.0 0.0.0.255
192.168.10.0 0.0.0.255 eq ftp-data
R1(config)# access-list 101 permit ip any any
R1(config)# interface g0/1
R1(config-if)#ip access-group 101 in
```

Configure Extended IPv4 ACLs

- Named extended ACLs are created in the same way that named standard ACLs are created.
- In this example, two named ACLs are created.
  - SURFING permits users on the 192.168.10.0/24 network to exit going to ports 80 and 443.
  - BROWSING enables return HTTP and HTTPs traffic.



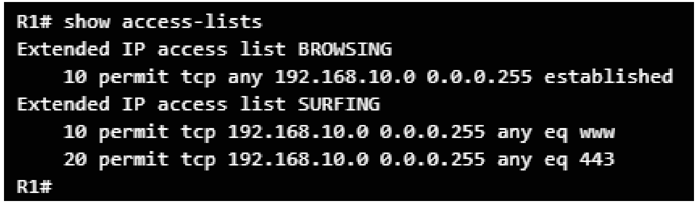
```
R1(config)# ip access-list extended SURFING
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config-ext-nacl)# exit
R1(config)# ip access-list extended BROWSING
R1(config-ext-nacl)# permit tcp any 192.168.10.0 0.0.0.255 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
```



Configure Extended IPv4 ACLs

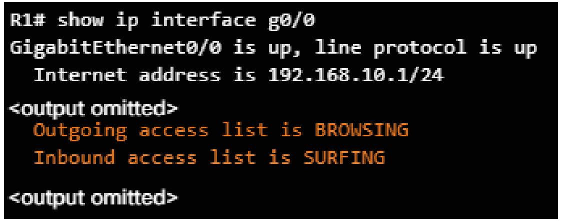
- The **show ip interface** and **show access-lists** commands can be used to verify the content of extended ACLs.

- The output and sequence numbers displayed in the **show access-lists** command output is the order in which the statements were entered.
  - Unlike standard ACLs, extended ACLs do not implement the same internal logic and hashing function.
  - Host entries are not automatically listed prior to range entries.



```
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
R1#
```

- The **show ip interface** command is used to verify the ACL on the interface and the direction in which it was applied.
  - The output from this command includes the number or name of the access list and the direction in which the ACL was applied.

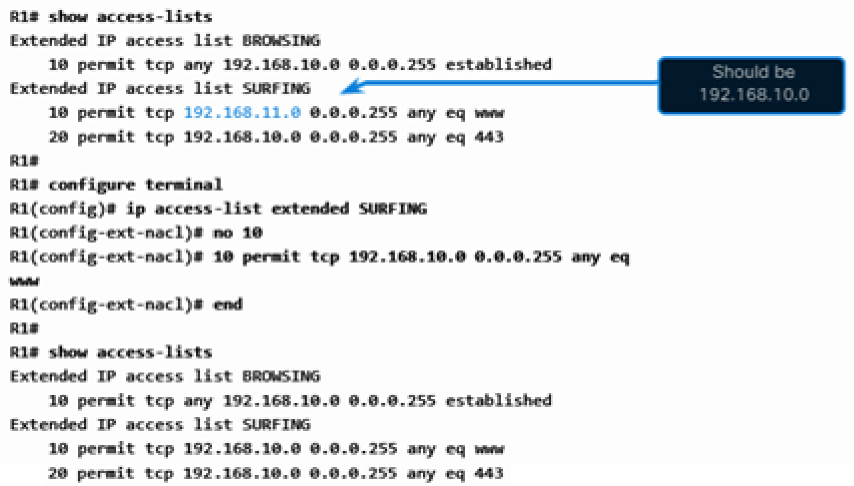


```
R1# show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet address is 192.168.10.1/24
<output omitted>
  Outgoing access list is BROWSING
  Inbound access list is SURFING
<output omitted>
```

Configure Extended IPv4 ACLs

- An extended ACL can be edited in one of two ways:
  - **Method 1 Text editor**
    - The ACL is copied and pasted into 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 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 this example, Method 2 is used to correct the named ACL SURFING which incorrectly permits 192.168.11.0/24 and is edited to permit 192.168.10.0/24.



```
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
```

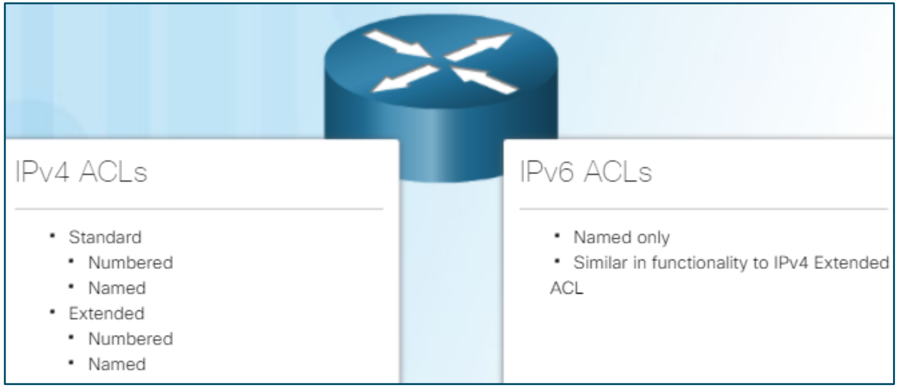
Should be 192.168.10.0

4.3 IPv6 ACLs

IPv6 ACL Creation

- IPv6 ACLs are similar to IPv4 ACLs in both operation and configuration.

In IPv4 there are two types of ACLs, standard and extended and both types of ACLs can be either numbered or named ACLs.



IPv4 ACLs

- Standard
  - Numbered
  - Named
- Extended
  - Numbered
  - Named

IPv6 ACLs

- Named only
- Similar in functionality to IPv4 Extended ACL

With IPv6, there is only one type of ACL, which is equivalent to an IPv4 extended named ACL and there are no numbered ACLs in IPv6.

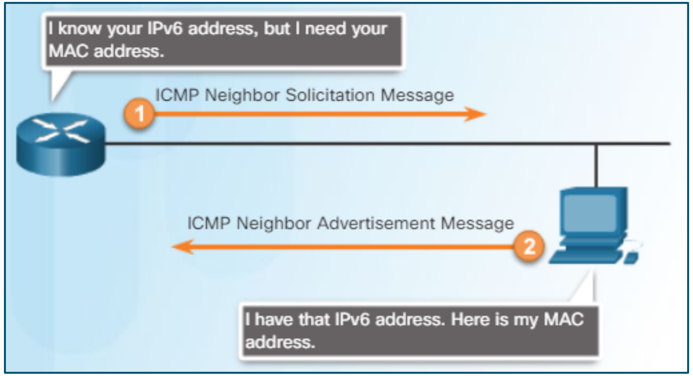
- **Note:**
  - An IPv4 ACL and an IPv6 ACL cannot share the same name.

IPv6 ACLs

IPv6 ACL Creation

- There are three significant differences between IPv4 and IPv6 ACLs:
  - The command used to apply an IPv6 ACL to an interface is **ipv6 traffic-filter** command.
  - IPv6 ACLs do not use wildcard masks but instead specifies the prefix-length to indicate how much of an IPv6 source or destination address should be matched.
  - An IPv6 ACL adds two implicit permit statements at the end of each IPv6 access list.
    - **permit icmp any any nd-na**
    - **permit icmp any any nd-ns**
    - **deny ipv6 any any statement**

- These two additional statements allow IPv6 ICMP Neighbor Discovery (ND) and Neighbor Solicitation (NS) messages to accomplish the same thing as IPv4 ARP.


I know your IPv6 address, but I need your MAC address.
1
ICMP Neighbor Solicitation Message
ICMP Neighbor Advertisement Message
2
I have that IPv6 address. Here is my MAC address.

4.4 Troubleshoot ACLs

Processing Packets with ACLs

- It is beneficial to consider how an inbound and outbound ACL is processed.
- Inbound ACLs operate as follows:
  - If the information in a packet header and an 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 and 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 and because of this statement, an ACL should have at least one permit statement in it; otherwise, the ACL blocks all traffic.
- Outbound ACLs operate as follows:
  - The router checks the routing table to see if the packet is routable.
  - The router checks to see whether the outbound interface is grouped to an ACL.
  - If it is, the ACL is tested by the combination of ACEs that are associated with that interface.
  - Based on the ACL tests, the packet is permitted or denied.

Processing Packets with ACLs

- When a packet arrives at a router interface:
  - The router checks to see whether the destination Layer 2 address matches its interface Layer 2 address.
  - If the frame is accepted, the router checks for an ACL on the inbound interface.
  - If an ACL exists, the packet is tested against the ACEs and the packet is either permitted or denied.
  - If the packet is permitted, it is then checked against routing table to determine the destination interface.
  - If a routing table entry exists for the destination, the packet is then switched to the outgoing interface.

- Next, the router checks whether the outgoing interface has an ACL.
  - If an ACL exists, the packet is tested against the ACEs.
  - If the packet matches an ACE, it is either permitted or denied.
  - If there is no ACL or the packet is permitted, the packet is encapsulated in the new Layer 2 protocol and forwarded out the interface to the next device.

Troubleshoot ACLs

Processing Packets with ACLs

- Standard ACLs only examine the source IPv4 address.
  - The destination of the packet and the ports involved are not considered.

- The Cisco IOS software tests addresses against the ACL ACEs.
  - The first match determines whether the software accepts or rejects the address.
  - Because the software stops testing conditions after the first match, the order of the conditions is critical.
  - If no conditions match, the address is rejected.

Troubleshoot ACLs

Processing Packets with ACLs

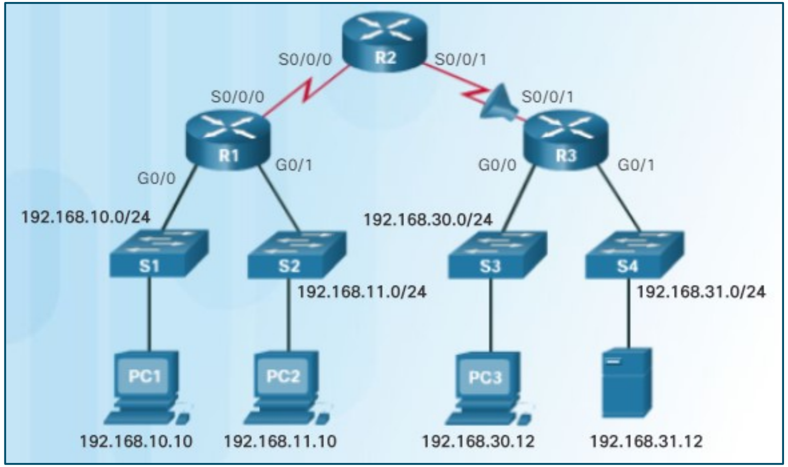
- Extended ACLs filter on protocol, source address, destination address, and port numbers.
- The ACL first filters on the source address, then on the port and protocol of the source.
- It then filters on the destination address, then on the port and protocol of the destination, and makes a final permit or deny decision.

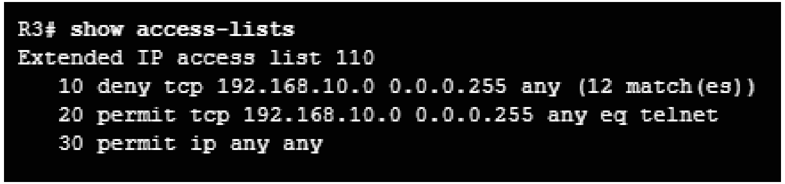
Common ACLs Errors

- The most common ACL errors are entering ACEs in the wrong order or not applying adequate criteria to the ACL rules.

- In this example, host 192.168.10.10 has no Telnet connectivity with 192.168.30.12.
  - The **show access-lists** command displays matches for the first deny statement indicating that this ACE has been matched by traffic.

- **Solution:**
  - Host 192.168.10.10 has no connectivity with 192.168.30.12 because statement 10 denies host 192.168.10.10, therefore statement 20 can never be matched.
  - Statements 10 and 20 should be reversed.



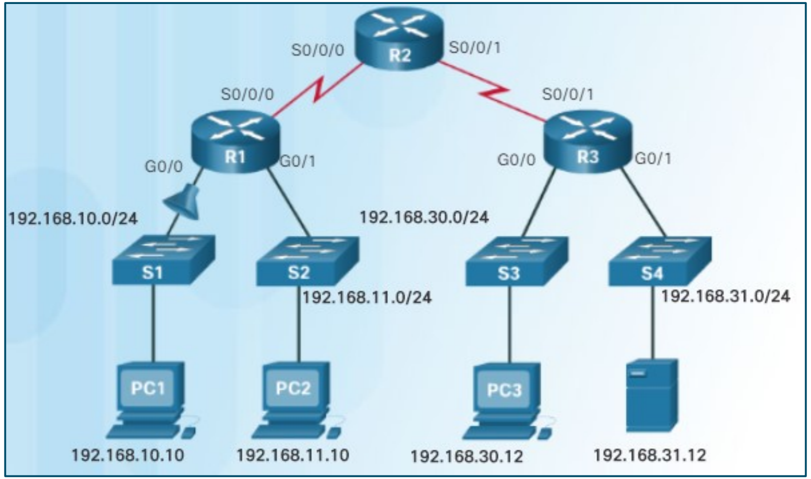


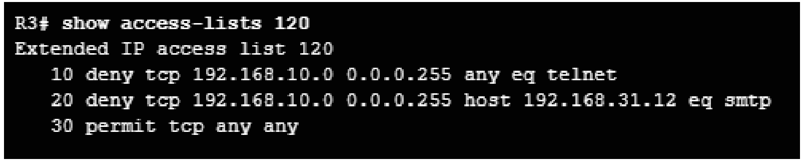
```
R3# show access-lists
Extended IP access list 110
   10 deny tcp 192.168.10.0 0.0.0.255 any (12 match(es))
   20 permit tcp 192.168.10.0 0.0.0.255 any eq telnet
   30 permit ip any any
```

Common ACLs Errors

- In this example, the 192.168.10.0/24 network cannot use TFTP to connect to the 192.168.30.0/24 network.

- **Solution:**
  - Statement 30 in access list 120 allows all TCP traffic.
  - However, TFTP uses UDP instead of TCP and therefore it is implicitly denied.
  - Statement 30 should be **permit ip any any**.



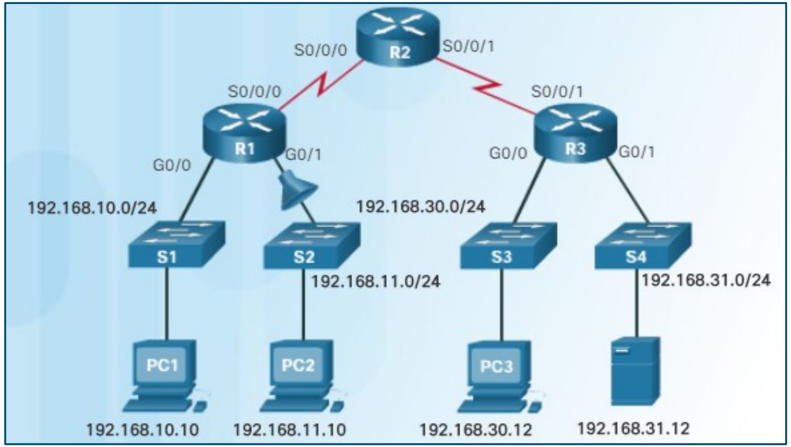


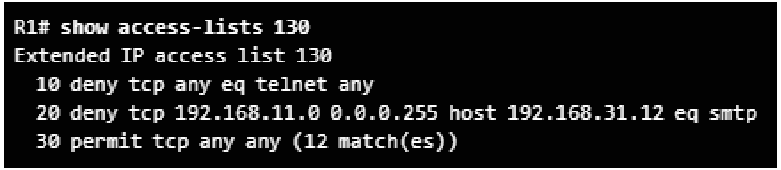
```
R3# show access-lists 120
Extended IP access list 120
   10 deny tcp 192.168.10.0 0.0.0.255 any eq telnet
   20 deny tcp 192.168.10.0 0.0.0.255 host 192.168.31.12 eq smtp
   30 permit tcp any any
```

Troubleshoot ACLs

Common ACLs Errors

- In this example, 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.



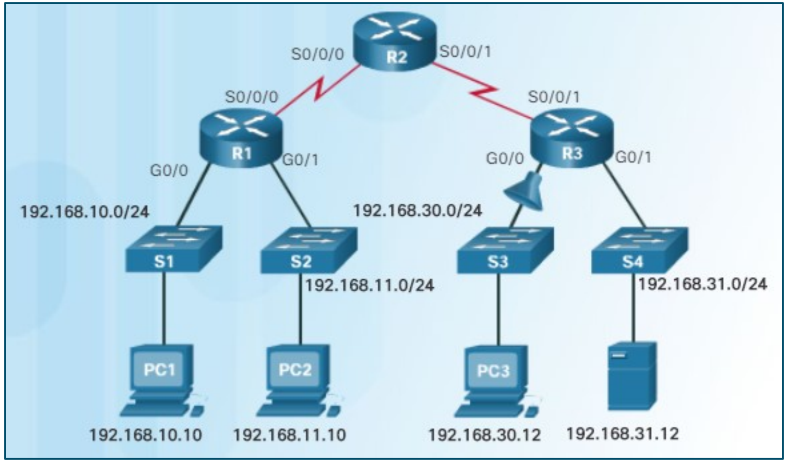


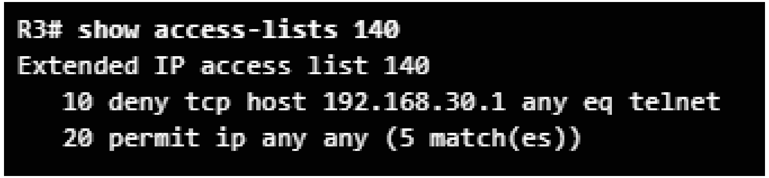
```
R1# show access-lists 130
Extended IP access list 130
  10 deny tcp any eq telnet any
  20 deny tcp 192.168.11.0 0.0.0.255 host 192.168.31.12 eq smtp
  30 permit tcp any any (12 match(es))
```

- **Solution:**
  - The Telnet port number in statement 10 of ACL 130 is listed in the wrong order as it currently denies any source packet with a port number equal to Telnet.
  - Configure **10 deny tcp 192.168.11.0 0.0.0.255 192.168.30.0 0.0.0.255 eq telnet**.

Common ACLs Errors

- In this example, 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.

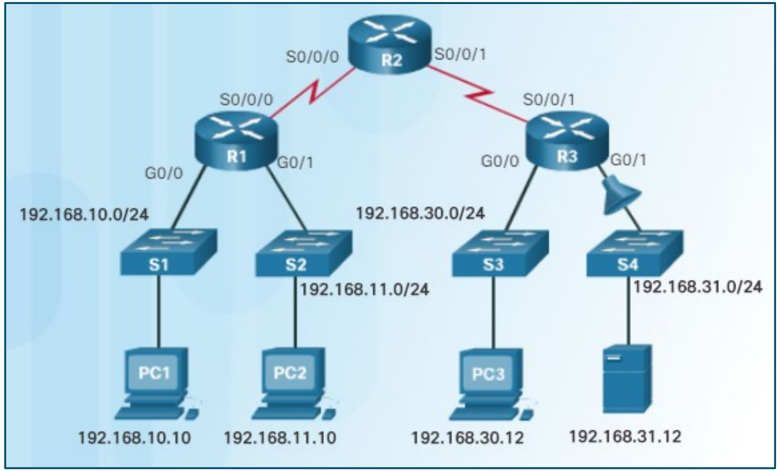


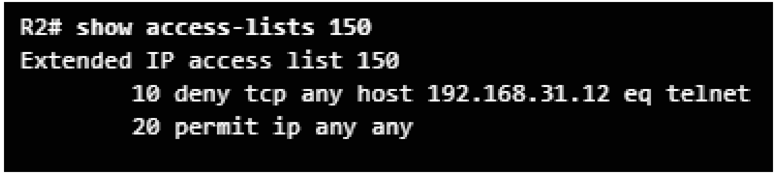


```
R3# show access-lists 140
Extended IP access list 140
    10 deny tcp host 192.168.30.1 any eq telnet
    20 permit ip any any (5 match(es))
```

Common ACLs Errors

- In this example, 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.



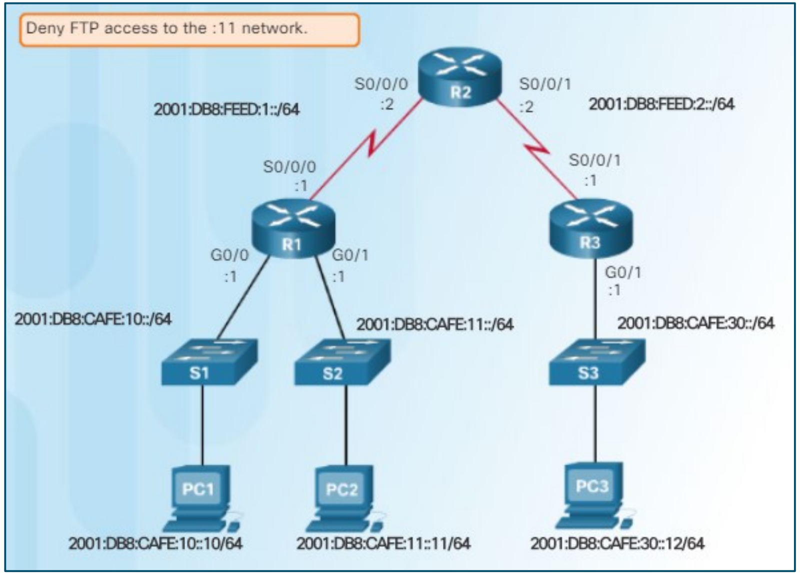


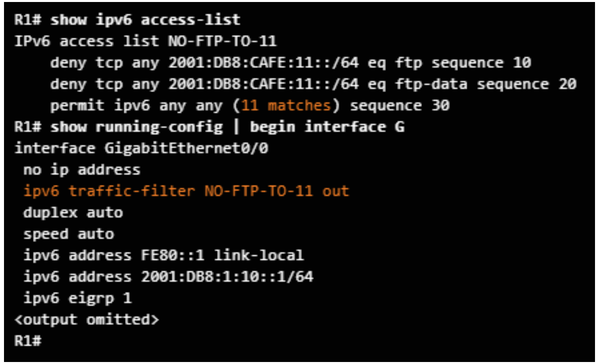
```
R2# show access-lists 150
Extended IP access list 150
        10 deny tcp any host 192.168.31.12 eq telnet
        20 permit ip any any
```

Troubleshoot ACLs

Common ACLs Errors

- In this example, R1 is configured with an IPv6 ACL to deny FTP access from the :10 network to the :11 network.
  - However, after configuring the ACL, PC1 is still able to connect to the FTP server running on PC2.
  - The output of the **show ipv6 access-list** command displays matches for the permit statement but not the deny statements.





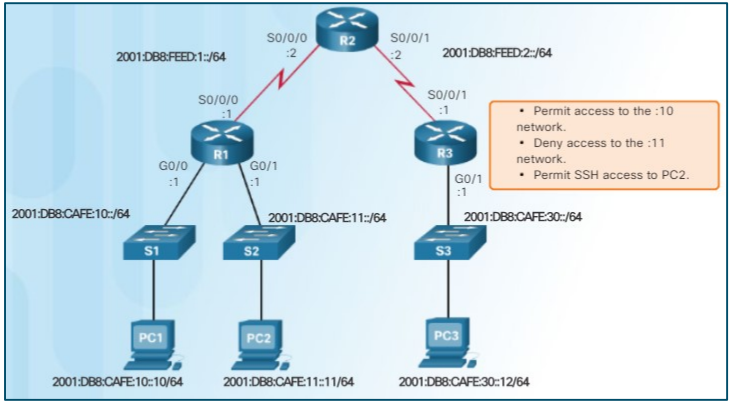
```
R1# show ipv6 access-list
IPv6 access list NO-FTP-TO-11
    deny tcp any 2001:DB8:CAFE:11::/64 eq ftp sequence 10
    deny tcp any 2001:DB8:CAFE:11::/64 eq ftp-data sequence 20
    permit ipv6 any any (11 matches) sequence 30
R1# show running-config | begin interface G
interface GigabitEthernet0/0
 no ip address
 ipv6 traffic-filter NO-FTP-TO-11 out
 duplex auto
 speed auto
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:1:10::1/64
 ipv6 eigrp 1
<output omitted>
R1#
```

- **Solution:**
  - The ACL was applied using the correct name, but not the correct direction.
  - To correct the issue, remove the **ipv6 traffic-filter NO-FTP-TO-11 out** and replace it with **ipv6 traffic-filter NO-FTP-TO-11 in**.

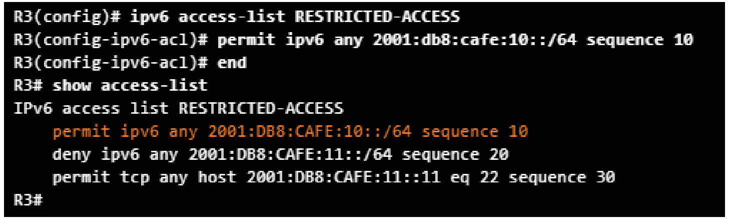
Troubleshoot ACLs

Common ACLs Errors

- In this example, R3 is configured with an IPv6 ACL named RESTRICTED-ACCESS that should permit access to the :10 network, deny access to the :11 network, and permit SSH access to the PC at 2001:DB8:CAFE:11::11
- After configuring the ACL, PC3 cannot reach the 10 or 11 network, and cannot SSH to 2001:DB8:CAFE:11::11.
- **Solution:**
  - The first permit statement should allow access to the :10 network but only access to the 2001:DB8:CAFE:10:: host is allowed.
  - To correct this issue, remove the host argument and change the prefix to /64. You can do this without removing the ACL by replacing the ACE using the sequence number 10.
  - The second error in the ACL is the order of the next two statements therefore remove the statements first, and then enter them in the correct order.



- Permit access to the :10 network.
- Deny access to the :11 network.
- Permit SSH access to PC2.

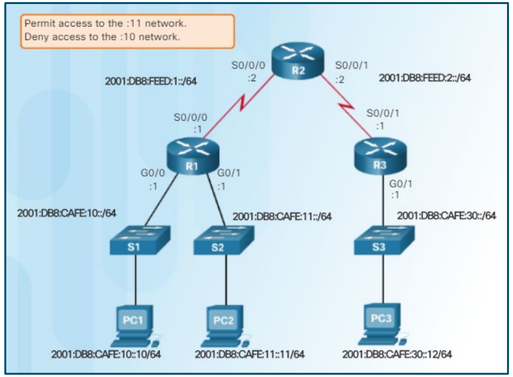


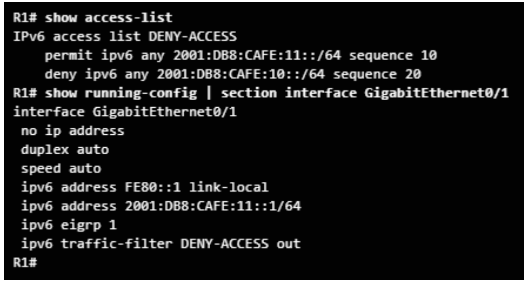
```
R3(config)# ipv6 access-list RESTRICTED-ACCESS
R3(config-ipv6-acl)# permit ipv6 any 2001:db8:cafe:10::/64 sequence 10
R3(config-ipv6-acl)# end
R3# show access-list
IPv6 access list RESTRICTED-ACCESS
    permit ipv6 any 2001:DB8:CAFE:10::/64 sequence 10
    deny ipv6 any 2001:DB8:CAFE:11::/64 sequence 20
    permit tcp any host 2001:DB8:CAFE:11::11 eq 22 sequence 30
R3#
```

Troubleshoot ACLs

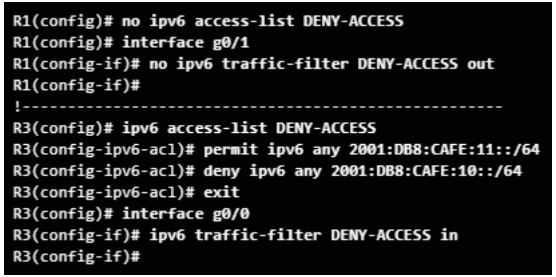
Common ACLs Errors

- In this example, R1 is configured with an IPv6 ACL named DENY-ACCESS that should permit access to the :11 network from the :30 network, but deny access to the :10 network.
  - The DENY-ACCESS ACL is supposed to permit access to the :11 network from the :30 network while denying access to the :10 network.
  - However, after applying the ACL to the interface the :10 network is still reachable from the :30 network.
- **Solution:**
  - The problem is with the location of the ACL and should be applied closest to the source of the traffic.
  - Remove the ACL on R1 and apply the ACL on R3.





```
R1# show access-list
IPv6 access list DENY-ACCESS
    permit ipv6 any 2001:DB8:CAFE:11::/64 sequence 10
    deny ipv6 any 2001:DB8:CAFE:10::/64 sequence 20
R1# show running-config | section interface GigabitEthernet0/1
interface GigabitEthernet0/1
 no ip address
 duplex auto
 speed auto
 ipv6 address FE80::1 link-local
 ipv6 address 2001:DB8:CAFE:11::1/64
 ipv6 eigrp 1
 ipv6 traffic-filter DENY-ACCESS out
R1#
```



```
R1(config)# no ipv6 access-list DENY-ACCESS
R1(config)# interface g0/1
R1(config-if)# no ipv6 traffic-filter DENY-ACCESS out
R1(config-if)#
!-----------------------------------------------------
R3(config)# ipv6 access-list DENY-ACCESS
R3(config-ipv6-acl)# permit ipv6 any 2001:DB8:CAFE:11::/64
R3(config-ipv6-acl)# deny ipv6 any 2001:DB8:CAFE:10::/64
R3(config-ipv6-acl)# exit
R3(config)# interface g0/0
R3(config-if)# ipv6 traffic-filter DENY-ACCESS in
R3(config-if)#
```

Access Control Lists

- By default a router does not filter traffic. Traffic that enters the router is routed solely based on information within the routing table.
- An ACL is a sequential list of permit or deny statements. The last statement of an ACL is always an implicit deny any statement which blocks all traffic. To prevent the implied deny any statement at the end of the ACL from blocking all traffic, the **permit ip any any** statement can be added.
- When network traffic passes through an interface configured with an ACL, the router compares the information within the packet against each entry, in sequential order, to determine if the packet matches one of the statements. If a match is found, the packet is processed accordingly.
- ACLs can be applied to inbound traffic or to outbound traffic.
- Standard ACLs can be used to permit or deny traffic only from a source IPv4 addresses. The basic rule for placing a standard ACL is to place it close to the destination.
- Extended ACLs filter packets based on several attributes: protocol type, source or destination IPv4 address, and source or destination ports. The basic rule for placing an extended ACL is to place it as close to the source as possible.

Access Control Lists (Cont.)

- The **access-list** global configuration command defines a standard ACL with a number in the range of 1 through 99 or an extended ACL with numbers in the range of 100 to 199. The **ip access-list standard** *name* is used to create a standard named ACL, whereas the command **ip access-list extended** *name* is for an extended access list.
- After an ACL is configured, it is linked to an interface using the **ip access-group** command in interface configuration mode. A device an only have one ACL per protocol, per direction, per interface.
- 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.
- The **show running-config** and **show access-lists** commands are used to verify ACL configuration. The **show ip interface** command is used to verify the ACL on the interface and the direction in which it was applied.
- The **access-class** command configured in line configuration mode is used to link an ACL to a particular VTY line.

Conclusion

Access Control Lists (Cont.)

- From global configuration mode, use the **ipv6 access-list** *name* command to create an IPv6 ACL. Unlike IPv4 ACLs, IPv6 ACLs do not use wildcard masks. Instead, the prefix-length is used to indicate how much of an IPv6 source or destination address should be matched.
- After an IPv6 ACL is configured, it is linked to an interface using the **ipv6 traffic-filter** command.
- Unlike IPv4, IPv6 ACLs do not have support for a standard or extended option.