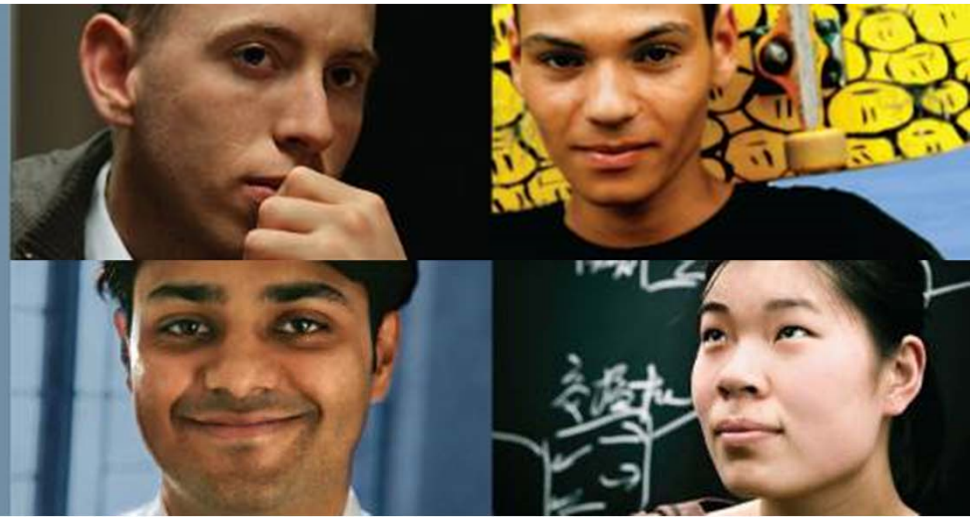
Chapter 9: Access Control Lists



Routing & Switching

Cisco | Networking Academy®
Mind Wide Open™

Chapter 9

9.1 IP ACL Operation

9.2 Standard IPv4 ACLs

9.3 Extended IPv4 ACLSs

9.4 Contextual Unit: Debug with ACLs

9.5 Troubleshoot ACLs

9.6 Contextual Unit: IPv6 ACLs

9.7 Summary

Chapter 9: Objectives

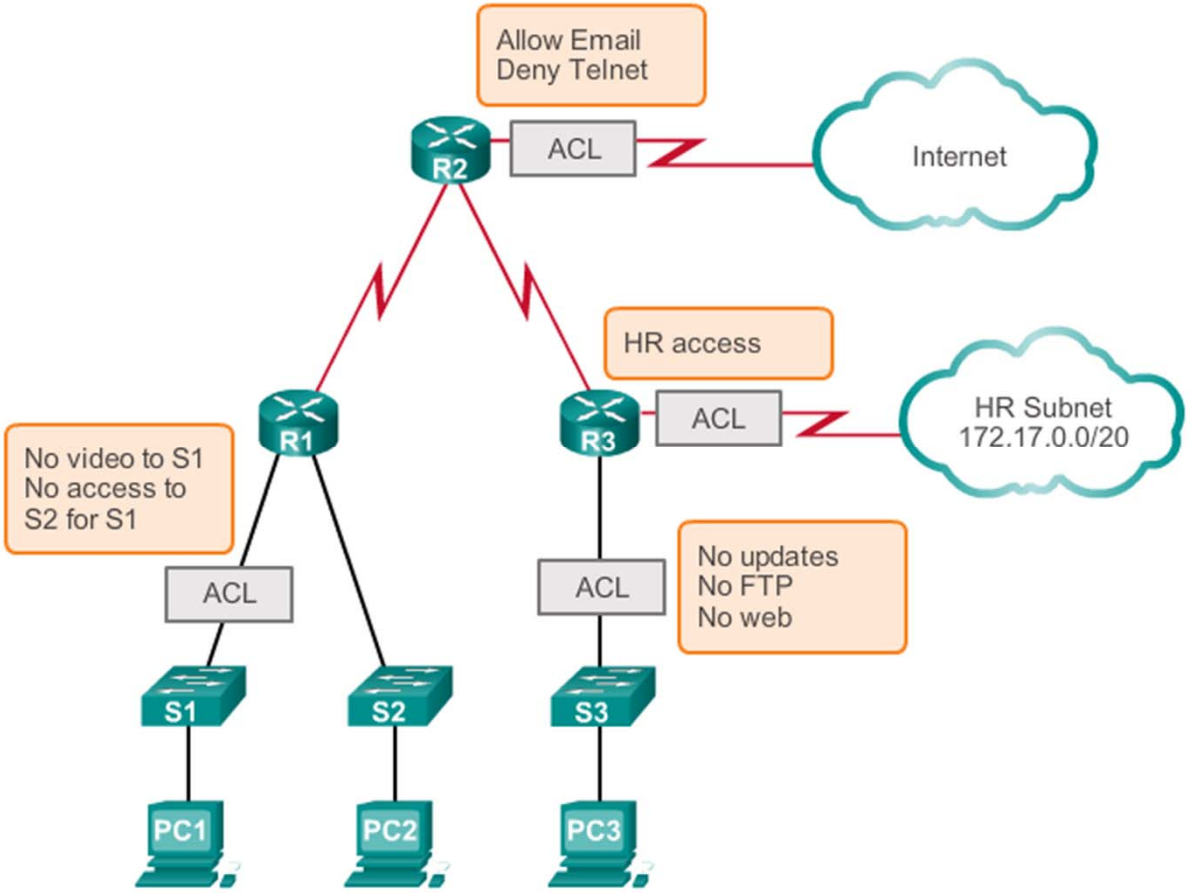
- Explain how ACLs are used to filter traffic.
- Compare standard and extended IPv4 ACLs.
- Explain how ACLs use wildcard masks.
- Explain the guidelines for creating ACLs.
- Explain the guidelines for placement of ACLs.
- Configure standard IPv4 ACLs to filter traffic according to networking requirements.
- Modify a standard IPv4 ACL using sequence numbers.
- Configure a standard ACL to secure vty access.

Chapter 9: Objectives (continued)

- Explain the structure of an extended access control entry (ACE).
- Configure extended IPv4 ACLs to filter traffic according to networking requirements.
- Configure an ACL to limit debug output.
- Explain how a router processes packets when an ACL is applied.
- Troubleshoot common ACL errors using CLI commands.
- Compare IPv4 and IPv6 ACL creation.
- Configure IPv6 ACLs to filter traffic according to networking requirements.

Purpose of ACLs

What is an ACL?



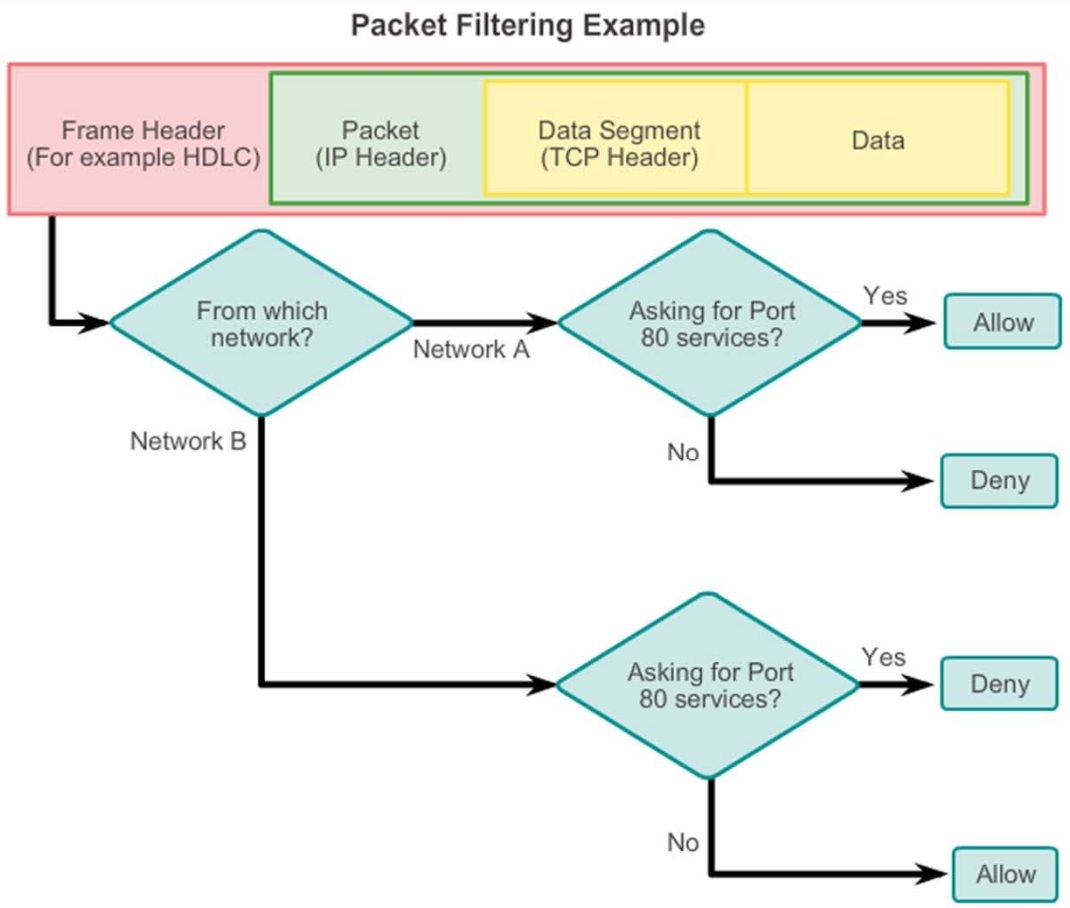
Purpose of ACLs

A TCP Conversation

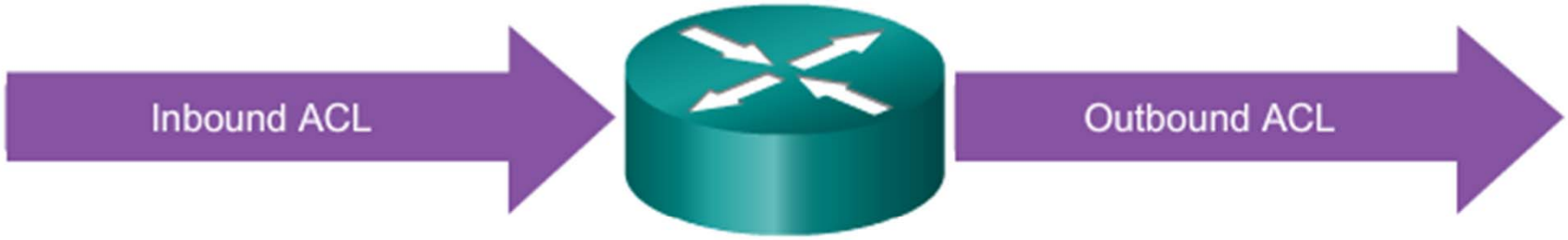
Packet Filtering

- Packet filtering, sometimes called static packet filtering, controls access to a network by analyzing the incoming and outgoing packets and passing or dropping them based on given criteria, such as the source IP address, destination IP addresses, and the protocol carried within the packet.
- A router acts as a packet filter when it forwards or denies packets according to filtering rules.
- An ACL is a sequential list of permit or deny statements, known as access control entries (ACEs).

Packet Filtering (Cont.)



ACL Operation



An inbound ACL filters packets coming into a specific interface and before they are routed to the outbound interface.

An outbound ACL filters packets after being routed, regardless of the inbound 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.

Types of Cisco IPv4 ACLs

Standard ACLs

```
access-list 10 permit 192.168.30.0 0.0.0.255
```

Standard ACLs filter IP packets based on the source address only.

Extended ACLs

```
access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq 80
```

Extended ACLs filter IP packets based on several attributes, including the following:

- Source and destination IP addresses
- Source and destination TCP and UDP ports
- Protocol type/ Protocol number (example: IP, ICP, UDP, TCP, etc.)

Standard versus Extended IPv4 ACLs

Numbering and Naming ACLs

Numbered ACL:

You assign a number based on which protocol you want filtered:

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

Named ACL:

You assign a name by providing the name of the ACL:

- Names can contain alphanumeric characters.
- It is suggested that the name be written in CAPITAL LETTERS.
- Names cannot contain spaces or punctuation.
- You can add or delete entries within the ACL.

Introducing ACL Wildcard Masking

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.

Wildcard masks are often referred to as an inverse mask. The reason is that, unlike a subnet mask in which binary 1 is equal to a match and binary 0 is not a match, in a wildcard mask the reverse is true.

Wildcard Masks in ACLs

Wildcard Mask Examples: Hosts / Subnets

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

Wildcard Masks in ACLs

Wildcard Mask Examples: Match Ranges

Example 1

	Decimal	Binary
IP Address	192.168.16.0	11000000.10101000.00010000.00000000
Wildcard Mask	0.0.15.255	00000000.00000000.00001111.11111111
Result Range	192.168.16.0 to 192.168.31.255	11000000.10101000.00010000.00000000 to 11000000.10101000.00011111.11111111

Example 2

	Decimal	Binary
IP Address	192.168.1.0	11000000.10101000.00000001.00000000
Wildcard Mask	0.0.254.255	00000000.00000000.11111110.11111111
Result	192.168.1.0	11000000.10101000.00000001.00000000
	All odd numbered subnets in the 192.168.0.0 major network	

Calculating the Wildcard Mask

Calculating wildcard masks can be challenging. One shortcut method is to subtract the subnet mask from 255.255.255.255.

Example 1

```
  255.255.255.255
- 255.255.255.000
-----------------
  000.000.000.255
```

Example 2

```
  255.255.255.255
- 255.255.255.240
-----------------
  000.000.000.015
```

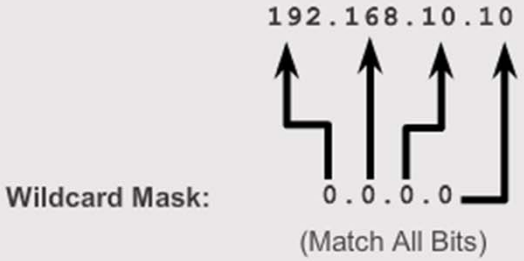
Example 3

```
  255.255.255.255
- 255.255.252.000
-----------------
  000.000.003.255
```

Wildcard Mask Keywords

Example 1

- 192.168.10.10 0.0.0.0 matches all of the address bits
- Abbreviate this wildcard mask using the IP address preceded by the keyword **host** (**host 192.168.10.10**)



Example 2

- 0.0.0.0 255.255.255.255 ignores all address bits
- Abbreviate expression with the keyword **any**

Examples Wildcard Mask Keywords

Example 1:

```
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 permit any
```

Example 2:

```
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
```

Guidelines for ACL creation

General Guidelines for Creating ACLs

- Use ACLs in firewall routers positioned between your internal network and an external network such as the Internet.
- Use ACLs on a router positioned between two parts of your network to control traffic entering or exiting a specific part of your internal network.
- Configure ACLs on border routers, that is routers situated at the edges of your networks.
- Configure ACLs for each network protocol configured on the border router interfaces.

General Guidelines for Creating ACLs (cont.)

The Three Ps

- 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.

Guidelines for ACL creation

ACL Best Practices

Guideline	Benefit
Base your ACLs on the security policy of the organization.	This will ensure you implement organizational security guidelines.
Prepare a description of what you want your ACLs to do.	This will help you avoid inadvertently creating potential access problems.
Use a text editor to create, edit and save ACLs.	This will help you create a library of reusable ACLs.
Test your ACLs on a development network before implementing them on a production network.	This will help you avoid costly errors.

Where to Place ACLs

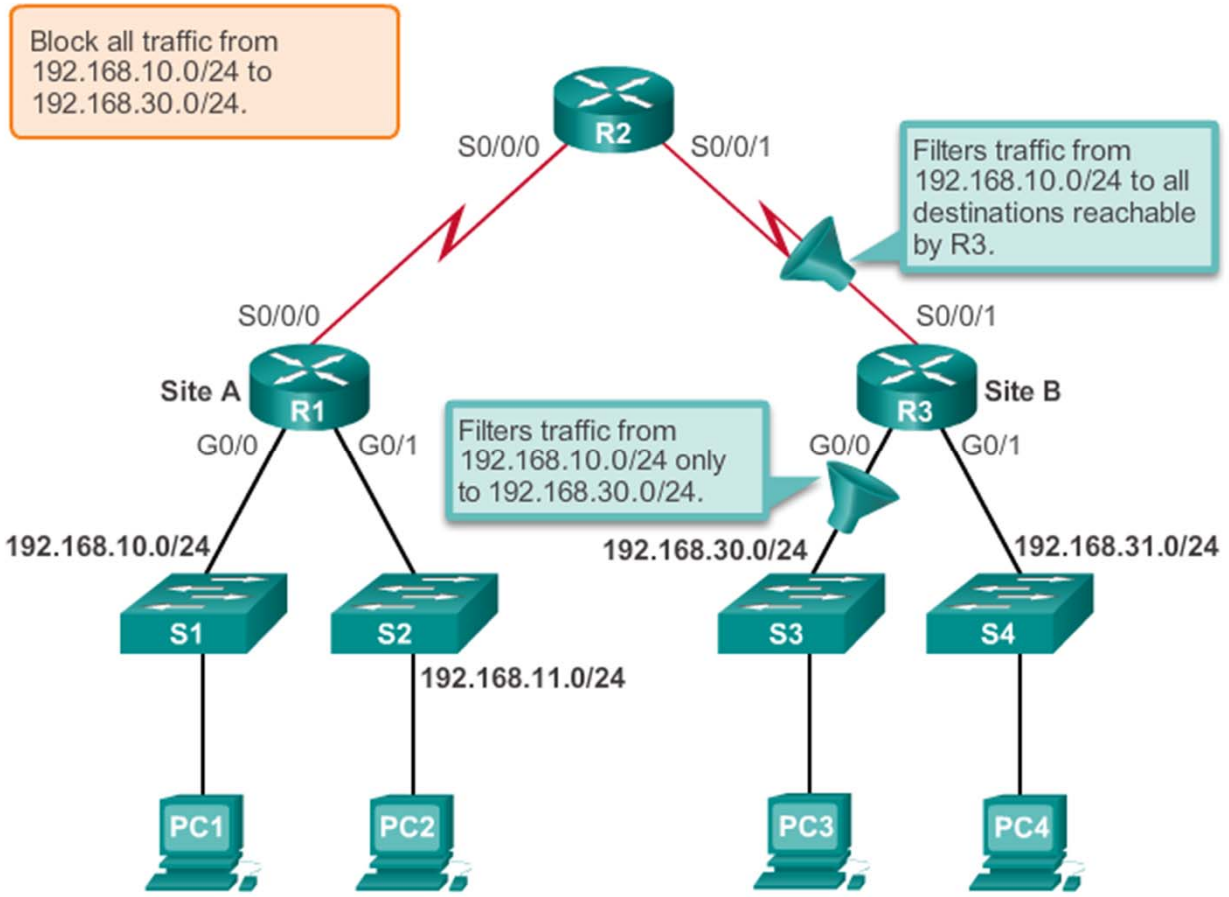
Every ACL should be placed where it has the greatest impact on efficiency. The basic rules are:

- Extended ACLs - Locate extended ACLs as close as possible to the source of the traffic to be filtered.
- Standard ACLs - Because standard ACLs do not specify destination addresses, place them as close to the destination as possible.

Placement of the ACL and therefore the type of ACL used may also depend on: the extent of the network administrator's control, bandwidth of the networks involved, and ease of configuration.

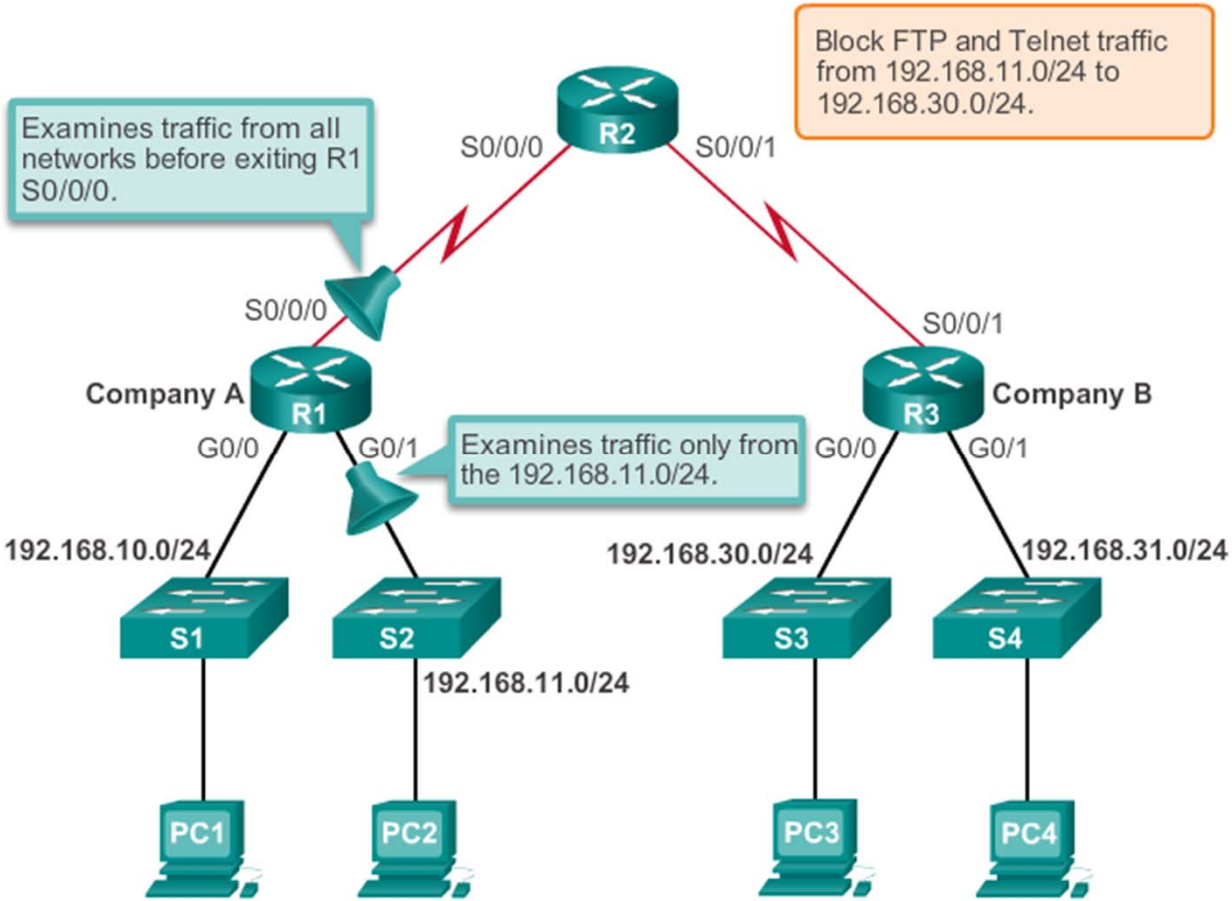
Guidelines for ACL Placement

Standard ACL Placement

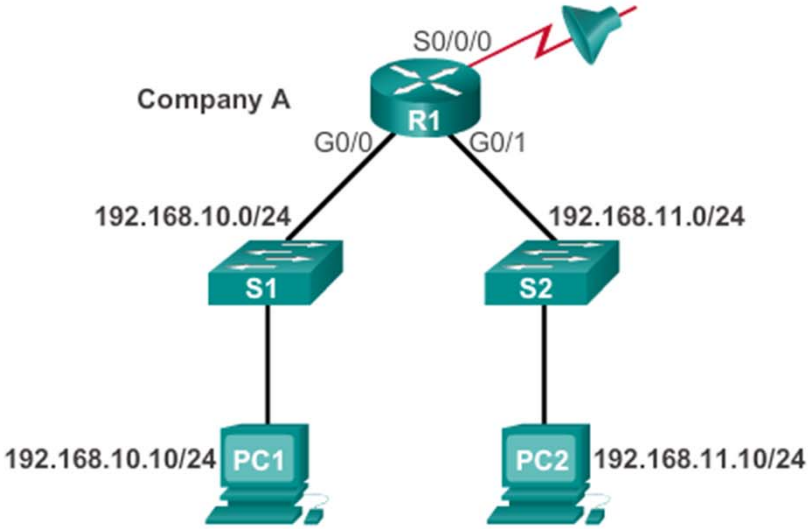


Guidelines for ACL Placement

Extended ACL Placement



Entering Criteria Statements



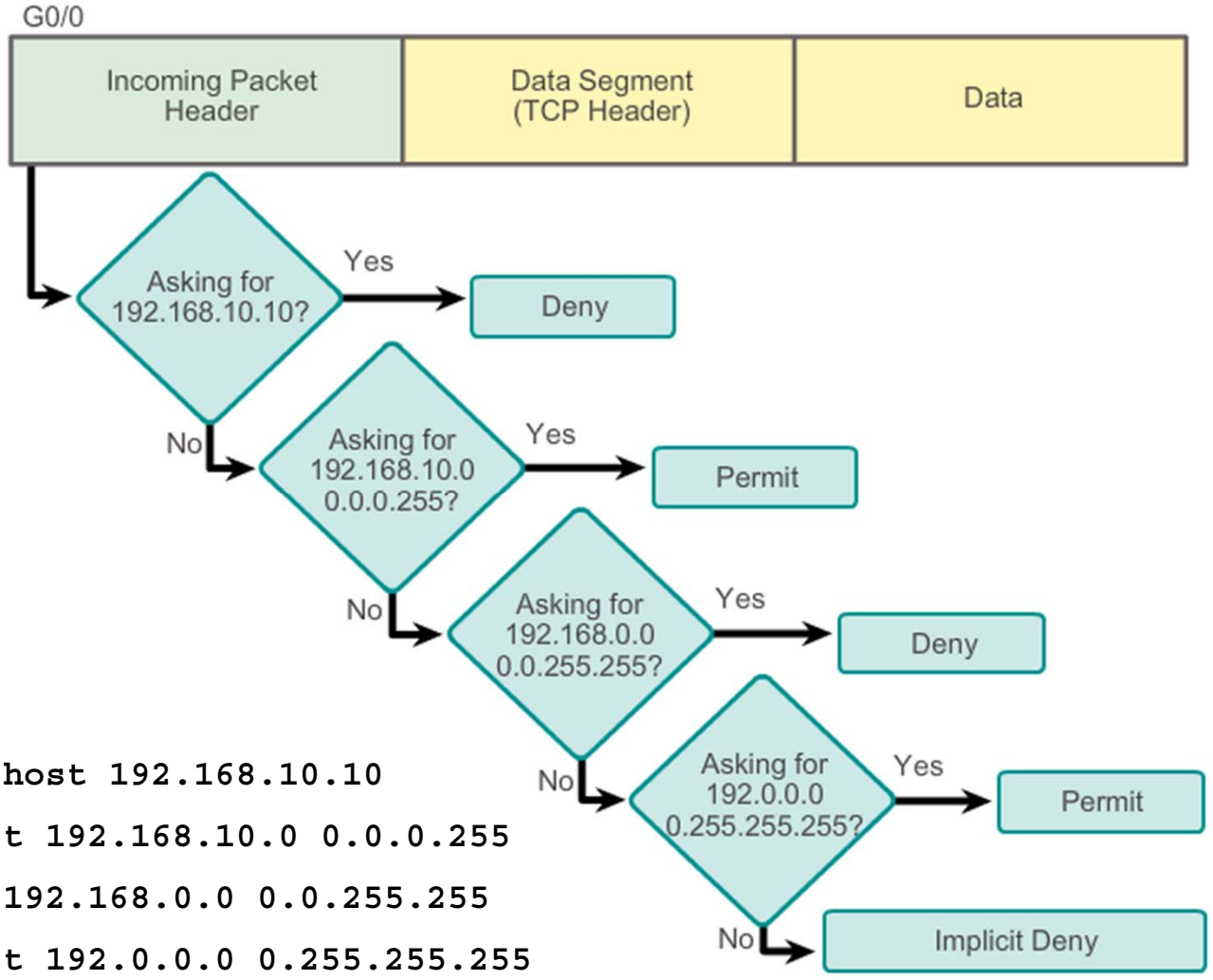
ACL 1

```
R1(config)#access-list 1 permit ip 192.168.10.0 0.0.0.255
```

ACL 2

```
R1(config)#access-list 2 permit ip 192.168.10.0 0.0.0.255
R1(config)#access-list 2 deny any
```

Configuring a Standard ACL



Example ACL

- `access-list 2 deny host 192.168.10.10`
- `access-list 2 permit 192.168.10.0 0.0.0.255`
- `access-list 2 deny 192.168.0.0 0.0.255.255`
- `access-list 2 permit 192.0.0.0 0.255.255.255`

Configuring a Standard ACL (cont.)

The full syntax of the standard ACL command is as follows:

```
Router(config)# access-list access-list-number
deny permit remark source [ source-wildcard ] [
log ]
```

To remove the ACL, the global configuration **no access-list** command is used.

The **remark** keyword is used for documentation and makes access lists a great deal easier to understand.

Internal Logic

- Cisco IOS applies an internal logic when accepting and processing standard access list statements. As discussed previously, access list statements are processed sequentially. Therefore, the order in which statements are entered is important.

```
R1(config)#access-list 3 deny 192.168.10.0 0.0.0.255
R1(config)#access-list 3 permit host 192.168.10.10
% Access rule can't be configured at higher sequence num as
it is part of the existing rule at sequence num 10
R1(config)#
```

ACL 3: Host statement conflicts with previous range statement.

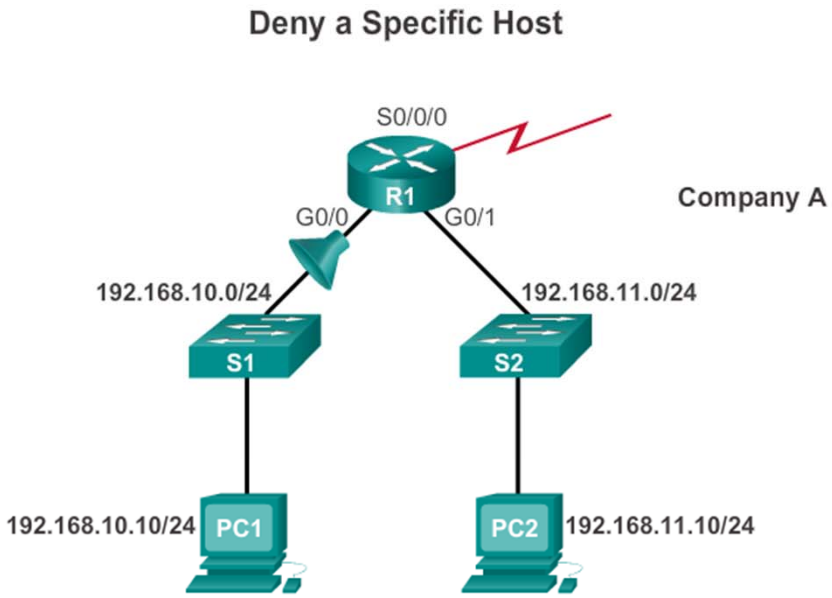
Applying Standard ACLs to Interfaces

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.

Applying Standard ACLs to Interfaces (Cont.)




```
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 permit any
R1(config)#interface g0/0
R1(config-if)#ip access-group 1 in
```

Creating Named Standard ACLs

```
Router(config)#ip access-list [standard | extended ] name
```

Alphanumeric name string must be unique and cannot begin with a number.

```
Router(config-std-nacl)#[permit | deny | remark] {source
[source- wildcard]} [log]
```

```
Router(config-if)#ip access-group name [in | out]
```

Activates the named IP ACL on an inteface.

Configure Standard IPv4 ACLs

Commenting ACLs

Example 1: Commenting a numbered ACL

```
R1(config)#access-list 1 remark Do not allow Guest workstation
through
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 remark Allow devices from all other
192.168.x.x subnets
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
R1(config)#interface s0/0/0
R1(config-if)#ip access-group 1 out
R1(config-if)#
```

Example 2: Commenting a named ACL

```
R1(config)#ip access-list standard NO_ACCESS
R1(config-std-nacl)#remark Do not allow access from Lab
workstation
R1(config-std-nacl)#deny host 192.168.11.10
R1(config-std-nacl)#remark Allow access from all other networks
R1(config-std-nacl)#permit any
R1(config-std-nacl)#interface G0/0
R1(config-if)#ip access-group NO_ACCESS out
R1(config-if)#
```

Modify IPv4 ACLs

Editing Standard Numbered ACLs

Editing Numbered ACLs Using a Text Editor

Configuration

```
R1(config)#access-list 1 deny host 192.168.10.99
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 1

```
R1#show running-config | include access-list 1
access-list 1 deny host 192.168.10.99
access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 2

```
<Text editor>
access-list 1 deny host 192.168.10.10
access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 3

```
R1#config t
Enter configuration commands, one per line.  End with
CNTL/Z.
R1(config)#no access-list 1
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 4

```
R1#show running-config | include access-list 1
access-list 1 deny host 192.168.10.10
access-list 1 permit 192.168.0.0 0.0.255.255
```

Editing Standard Numbered ACLs (cont.)

Editing Numbered ACLs Using Sequence Numbers

Configuration

```
R1(config)#access-list 1 deny host 192.168.10.99
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 1

```
R1#show access-lists 1
Standard IP access list 1
    10 deny   192.168.10.99
    20 permit 192.168.0.0, wildcard bits 0.0.255.255
R1#
```

Step 2

```
R1#conf t
R1(config)#ip access-list standard 1
R1(config-std-nacl)#no 10
R1(config-std-nacl)#10 deny host 192.168.10.10
R1(config-std-nacl)#end
R1#
```

Step 3

```
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
R1#
```

Modify IPv4 ACLs

Editing Standard Named ACLs

Adding a Line to a Named ACL

```
R1#show access-lists
Standard IP access list NO_ACCESS
    10 deny   192.168.11.10
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#conf t
Enter configuration commands, one per line.  End with
CNTL/Z.
R1(config)#ip access-list standard NO_ACCESS
R1(config-std-nacl)#15 deny host 192.168.11.11
R1(config-std-nacl)#end
R1#show access-lists
Standard IP access list NO_ACCESS
    10 deny   192.168.11.10
    15 deny   192.168.11.11
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

Note: The **no sequence-number** named-ACL command is used to delete individual statements.

Modify IPv4 ACLs

Verifying ACLs

```
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#
```

Modify IPv4 ACLs

ACL Statistics

```
R1#show access-lists
Standard IP access list 1
    10 deny   192.168.10.10 (4 match(es))
    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 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

Output after pinging PC3 from PC1.

```
R1#show access-lists
Standard IP access list 1
    10 deny   192.168.10.10 (8 match(es))
    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 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

Matches have been incremented.

Standard ACL Sequence Numbers

- Another part of the IOS internal logic involves the internal sequencing of standard ACL statements. Range statements that deny three networks are configured first followed by five host statements. The host statements are all valid statements because their host IP addresses are not part of the previously entered range statements.
- The host statements are listed first by the show command, but not necessarily in the order that they were entered. The IOS puts host statements in an order using a special hashing function. The resulting order optimizes the search for a host ACL entry.

Configuring a Standard ACL to Secure a VTY Port

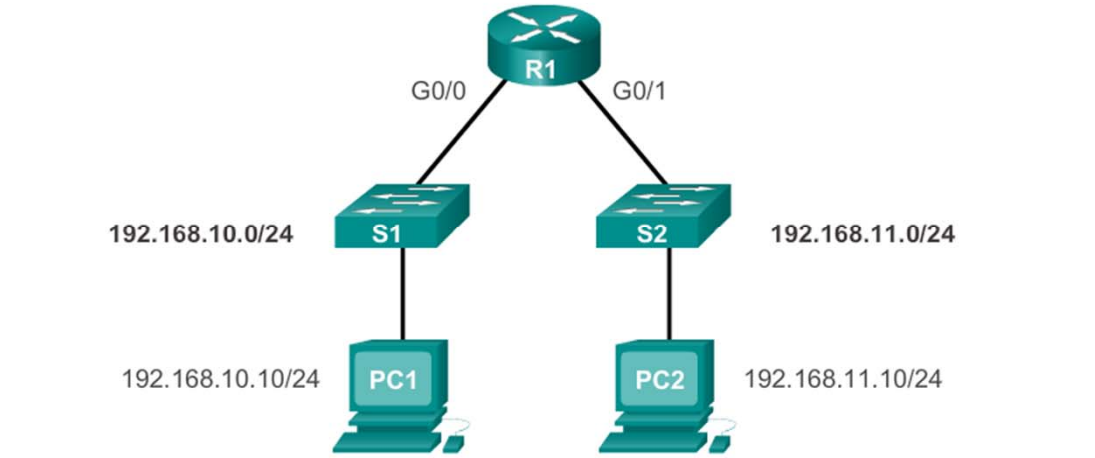
Filtering Telnet or SSH traffic is typically considered an extended IP ACL function because it filters a higher level protocol. However, because the **access-class** command is used to filter incoming or outgoing Telnet/SSH sessions by source address, a standard ACL can be used.

```
Router(config-line)# access-class access-list-number { in [ vrf-also ] | out }
```

Securing VTY ports with a Standard IPv4 ACL

Verifying a Standard ACL used to Secure a VTY Port

```
R1#show access-lists
Standard IP access list 21
    10 permit 192.168.10.0, wildcard bits 0.0.0.255 (2 matches)
    20 deny   any (1 match)
R1#
```




```
PC1>ssh 192.168.10.1

Login as: admin
Password: *****
R1>
```

```
PC2>ssh 192.168.11.1
ssh connect to host 192.168.11.1 port
22: Connection refused

PC2>
```

Structure of an Extended IPv4 ACL

Extended ACLs

Extended ACLs can filter on:

- Source address
- Destination address
- Protocol
- Port numbers

Structure of an Extended IPv4 ACL

Extended ACLs (Cont.)

Using Port Numbers

```
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
```

Using Keywords

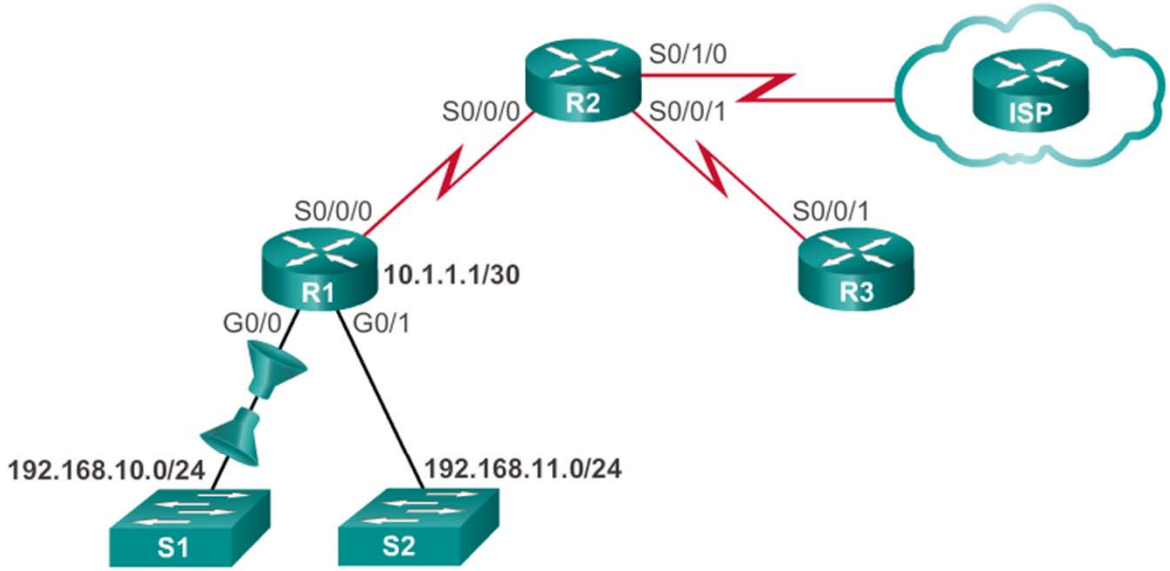
```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq telnet
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp-data
```

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.

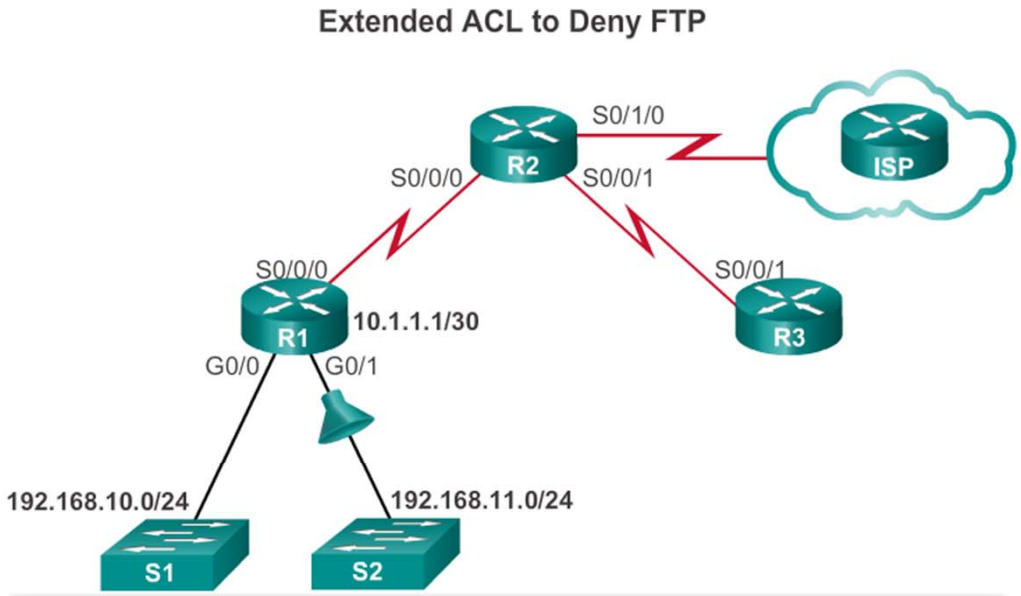
```
access-list access-list-number {deny | permit | remark}
protocol source [source-wildcard] [operator operand]
[port port-number or name] destination [destination-wildcard]
[operator operand] [port port-number or name][established]
```

Applying Extended ACLs to Interfaces



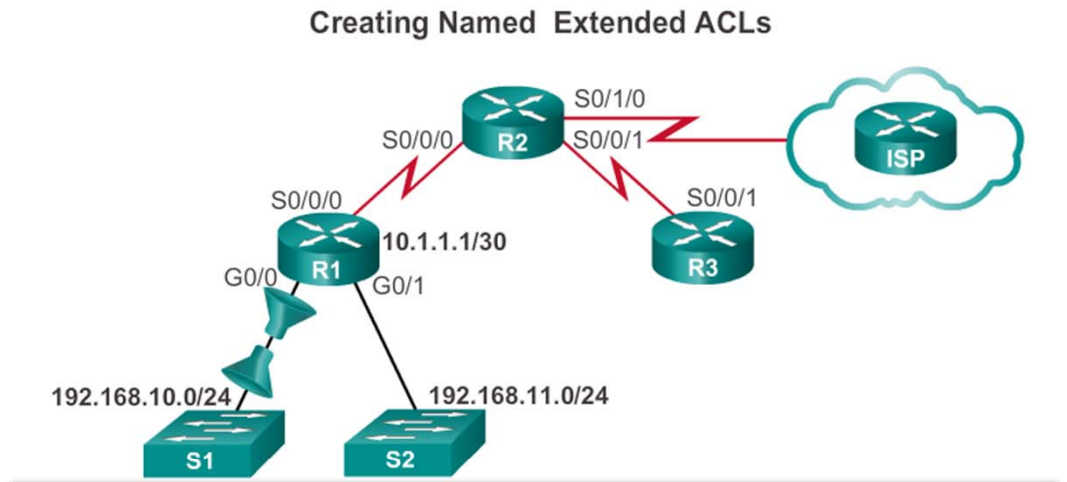

```
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)#access-list 104 permit tcp any 192.168.10.0 0.0.0.255 established
R1(config)#interface g0/0
R1(config-if)#ip access-group 103 in
R1(config-if)#ip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

Filtering Traffic with Extended ACLs




```
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
```

Creating Named Extended ACLs




```
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
```

Verifying Extended ACLs

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

Editing Extended ACLs

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

- Method 1 - Text editor
- Method 2 – Sequence numbers

Inbound ACL Logic

- Packets are tested against an inbound ACL, if one exists, before being routed.
- If an inbound packet matches an ACL statement with a permit, it is sent to be routed.
- If an inbound packet matches an ACL statement with a deny, it is dropped and not routed.
- If an inbound packet does not meet any ACL statements, then it is “implicitly denied” and dropped without being routed.

Outbound ACL Logic

- Packets are first checked for a route before being sent to an outbound interface. If there is no route, the packets are dropped.
- If an outbound interface has no ACL, then the packets are sent directly to that interface.
- If there is an ACL on the outbound interface, it is tested before being sent to that interface.
- If an outbound packet matches an ACL statement with a permit, it is sent to the interface.

Outbound ACL Logic (cont.)

- If an outbound packet matches an ACL statement with a deny, it is dropped.
- If an outbound packet does not meet any ACL statements, then it is “implicitly denied” and dropped.

ACL Logic Operations

- When a packet arrives at a router interface, the router process is the same, whether ACLs are used or not. As a frame enters an interface, the router checks to see whether the destination Layer 2 address matches its the interface Layer 2 address or if the frame is a broadcast frame.
- If the frame address is accepted, the frame information is stripped off and the router checks for an ACL on the inbound interface. If an ACL exists, the packet is tested against the statements in the list.

ACL Logic Operations (cont.)

- If the packet is accepted, it is then checked against routing table entries to determine the destination interface. If a routing table entry exists for the destination, the packet is then switched to the outgoing interface, otherwise the packet is dropped.
- Next, the router checks whether the outgoing interface has an ACL. If an ACL exists, the packet is tested against the statements in the list.
- 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.

Standard ACL Decision Process

- Standard ACLs only examine the source IPv4 address. The destination of the packet and the ports involved are not considered.
- Cisco IOS software tests addresses against the conditions in the ACL. 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.

Extended ACL Decision Process

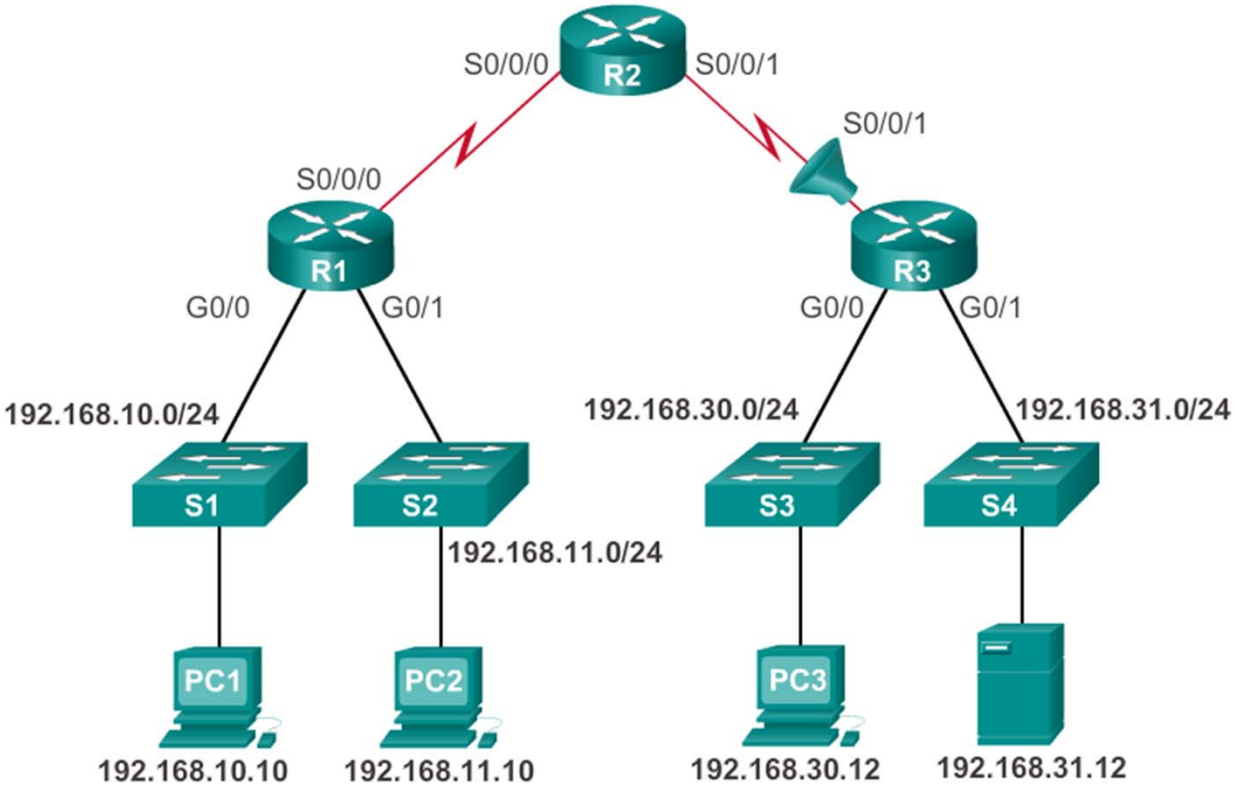
The ACL first filters on the source address, then on the port and protocol of the source. It then filters on the destination address, then on the port and protocol of the destination, and makes a final permit or deny decision.

Troubleshooting Common ACL Errors - Example 1

Host 192.168.10.10

has no connectivity

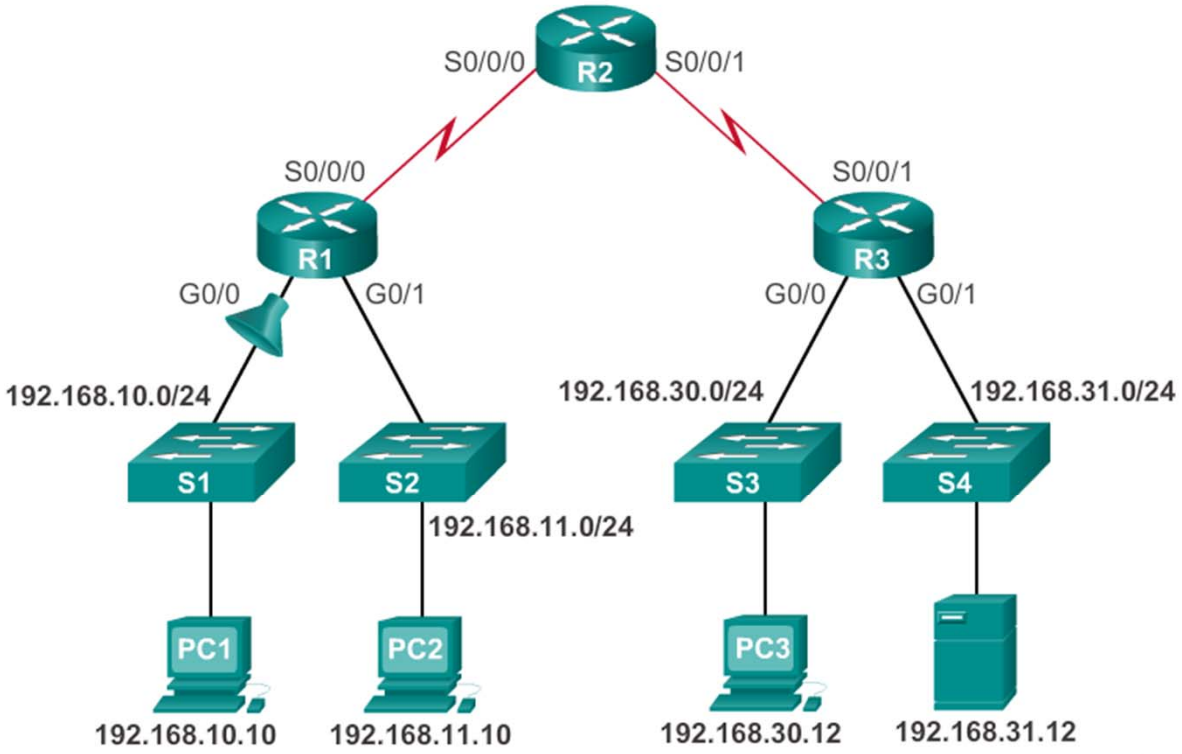
with 192.168.30.12.




```
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
```

Troubleshooting Common ACL Errors – Example 2

The 192.168.10.0 /24 network cannot use TFTP to connect to the 192.168.30.0 /24 network.

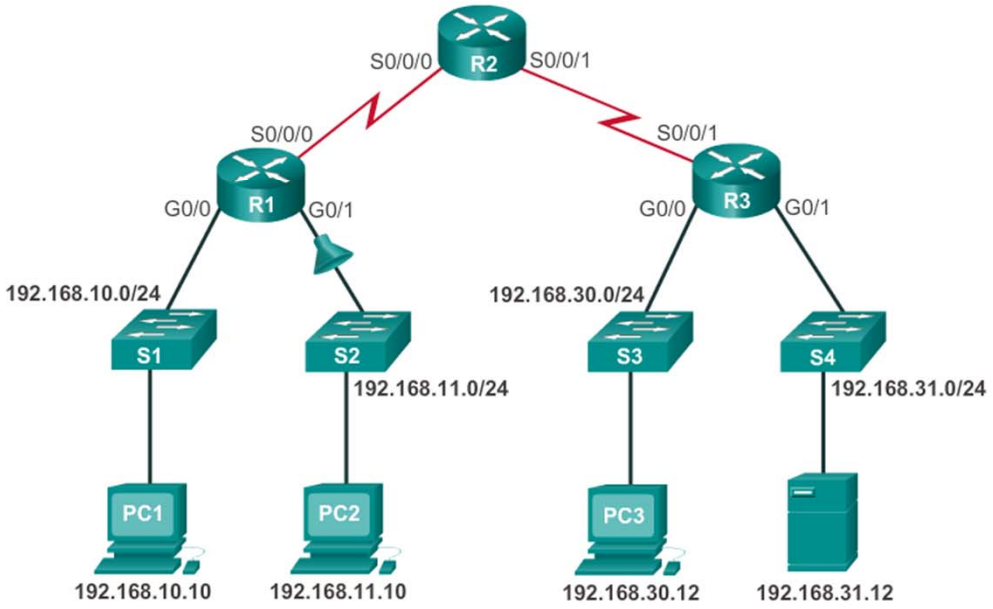



```
R1#show access-lists 120
Extended IP access list 120
    10 deny tcp 192.168.10.0 0.0.0.255 any eq telnet
    20 deny tcp 192.168.10.0 0.0.0.255 host 192.168.31.12 eq smtp
    30 permit tcp any any
```

Troubleshooting Common ACL Errors – Example 3

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.

```
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))
```

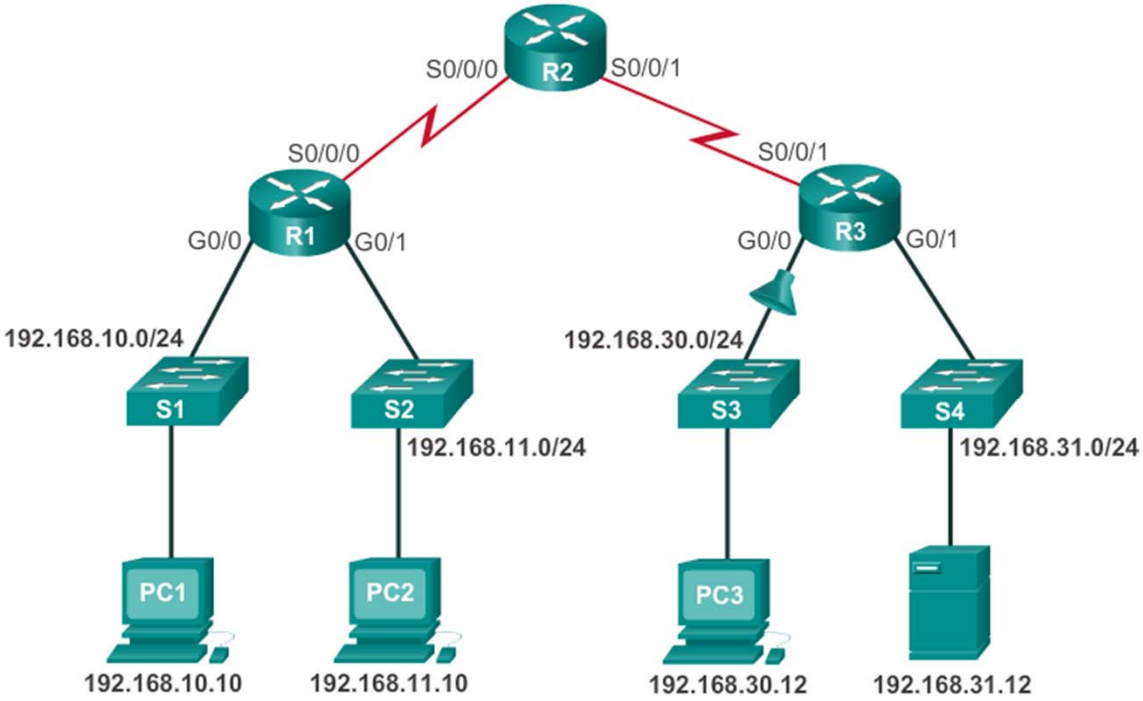


Common ACLs Errors

Troubleshooting Common ACL Errors – Example 4

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.

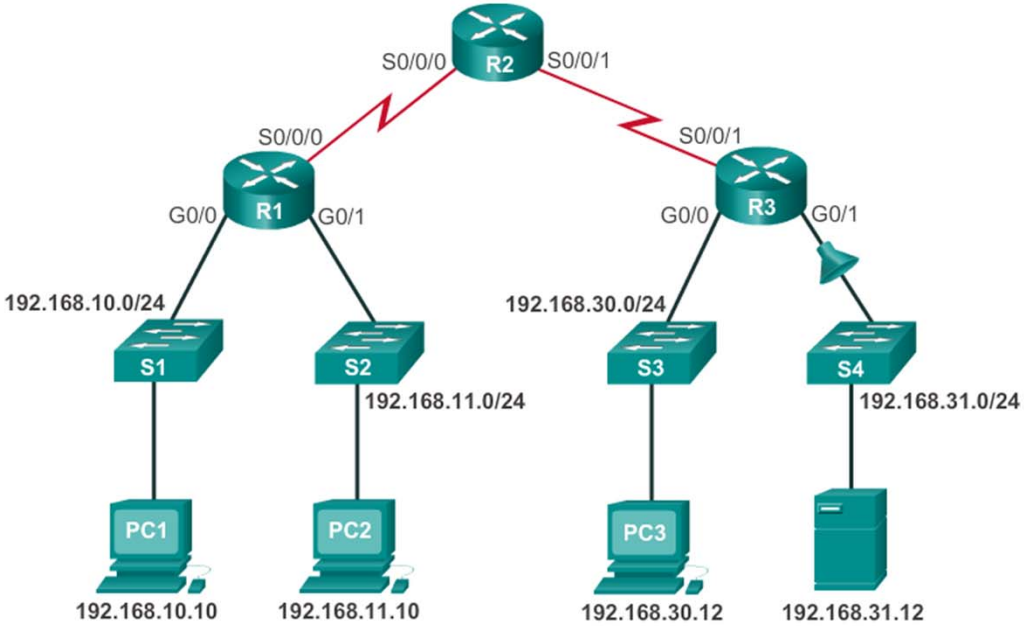
```
R3#show access-lists 140
Extended IP access list 140
    10 deny tcp host 192.168.30.1 any eq telnet
    20 permit ip any any (5 match(es))
```



Troubleshooting Common ACL Errors – Example 5

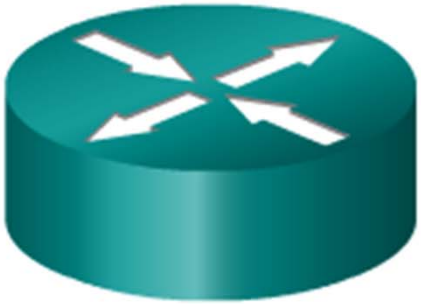
Host 192.168.30.12 can use Telnet to connect to 192.168.31.12, but according to the security policy, this connection should not be allowed.

```
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
```



IPv6 ACL Creation

Type of IPv6 ACLs



IPv4 ACLs

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

IPv6 ACLs

- Named only
- Similar in functionality to IPv4 Extended ACL

Comparing IPv4 and IPv6 ACLs

Although IPv4 and IPv6 ACLs are very similar, there are three significant differences between them.

- Applying an IPv6 ACL

  IPv6 uses the **`ipv6 traffic-filter`** command to perform the same function for IPv6 interfaces.

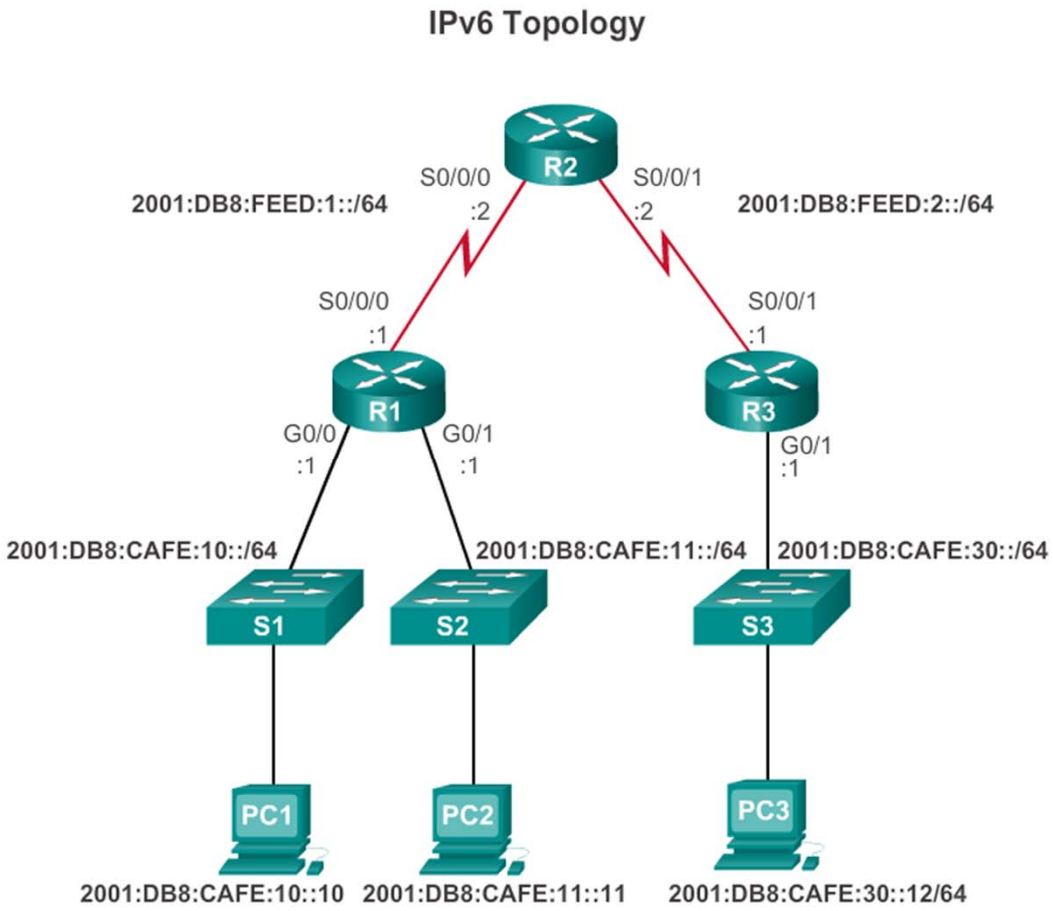
- No Wildcard Masks

  The prefix-length is used to indicate how much of an IPv6 source or destination address should be matched.

- Additional Default Statements

```
permit icmp any any nd-na
permit icmp any any nd-ns
```

Configuring IPv6 Topology



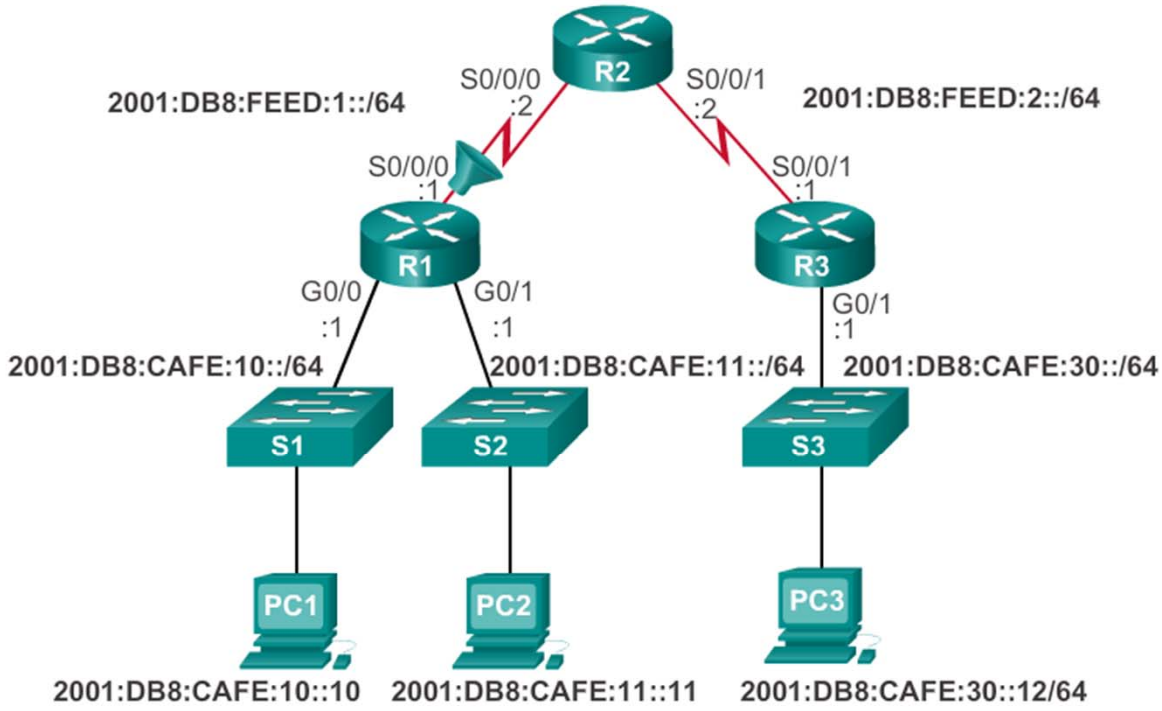
Configuring IPv6 ACLs

There are three basic steps to configure an IPv6 ACL:

1. From global configuration mode, use the **ipv6 access-list** *name* command to create an IPv6 ACL.
2. From the named ACL configuration mode, use the **permit** or **deny** statements to specify one or more conditions to determine if a packet is forwarded or dropped.
3. Return to privileged EXEC mode with the **end** cor

```
R1(config)# ipv6 access-list access-list-name
R1(config-ipv6-acl)# deny | permit protocol {source-ipv6-
prefix/prefix-length | any | host source-ipv6-address} [operator
[port-number]] {destination-ipv6-prefix/ prefix-length | any |
host destination-ipv6-address} [operator [port-number]]
```

Applying an IPv6 ACL to an Interface




```
R1(config)#interface s0/0/0
R1(config-if)#ipv6 traffic-filter NO-R3-LAN-ACCESS in
```

IPv6 ACL Examples

Deny FTP

```
R1(config)#ipv6 access-list NO-FTP-TO-11
R1(config-ipv6-acl)#deny tcp any 2001:db8:cafe:11::/64 eq ftp
R1(config-ipv6-acl)#deny tcp any 2001:db8:cafe:11::/64 eq ftp-data
R1(config-ipv6-acl)#permit ipv6 any any
R1(config-ipv6-acl)#exit
R1(config)#interface g0/0
R1(config-if)#ipv6 traffic-filter NO-FTP-TO-11 in
R1(config-if)#
```

Restrict Access

```
R3(config)#ipv6 access-list RETRICTED-ACCESS
R3(config-ipv6-acl)#remark Permit access only HTTP and HTTPS to Network 1
R3(config-ipv6-acl)#permit tcp any host 2001:db8:cafe:10::10 eq 80
R3(config-ipv6-acl)#permit tcp any host 2001:db8:cafe:10::10 eq 443   1
R3(config-ipv6-acl)#remark Deny all other traffic to Network 10
R3(config-ipv6-acl)#deny ipv6 any 2001:db8:cafe:10::/64   2
R3(config-ipv6-acl)#remark Permit PC3 telnet access to PC2
R3(config-ipv6-acl)#permit tcp host 2001:DB8:CAFE:30::12 host 2001:DB8:CA
R3(config-ipv6-acl)#remark Deny telnet access to PC2 for all other device
R3(config-ipv6-acl)#deny tcp any host 2001:db8:cafe:11::11 eq 23   4
R3(config-ipv6-acl)#remark Permit access to everything else
R3(config-ipv6-acl)#permit ipv6 any any   5
R3(config-ipv6-acl)#exit
R3(config)#interface g0/0
R3(config-if)#ipv6 traffic-filter RESTRICTED-ACCESS in   6
R3(config-if)#
```

Configuring IPv6 ACLs

Verifying IPv6 ACLs

```
R3#show ipv6 interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Global unicast address(es):
    2001:DB8:CAFE:30::1, subnet is 2001:DB8:CAFE:30::/64
  Input features: Access List
  Inbound access list RESTRICTED-ACCESS
<some output omitted for brevity>
```

```
R3#show access-lists
IPv6 access list RESTRICTED-ACCESS
    permit tcp any host 2001:DB8:CAFE:10::10 eq www sequence 20
    permit tcp any host 2001:DB8:CAFE:10::10 eq 443 sequence 30
    deny ipv6 any 2001:DB8:CAFE:10::/64 sequence 50
    permit tcp host 2001:DB8:CAFE:30::12 host 2001:DB8:CAFE:11::11 eq
    telnet sequence 70
    deny tcp any host 2001:DB8:CAFE:11::11 eq telnet sequence 90
    permit ipv6 any any sequence 110
R3#
```

Chapter 9: Summary

- By default a router does not filter traffic. Traffic that enters the router is routed solely based on information within the routing table.
- Packet filtering, controls access to a network by analyzing the incoming and outgoing packets and passing or dropping them based on criteria such as the source IP address, destination IP addresses, and the protocol carried within the packet.
- A packet-filtering router uses rules to determine whether to permit or deny traffic. A router can also perform packet filtering at Layer 4, the transport layer.
- An ACL is a sequential list of permit or deny statements.

Chapter 9: Summary (cont.)

- The last statement of an ACL is always an implicit deny 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 are configured to apply to inbound traffic or to apply to outbound traffic.

Chapter 9: Summary (cont.)

- 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 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.

Chapter 9: Summary (cont.)

- The **access-list** global configuration command defines a standard ACL with a number in the range of 1 to 99 or an extended ACL with numbers in the range of 100 to 199 and 2000 to 2699. Both standard and extended ACLs can be named.
- 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. IPv4 ACL statements include the use of wildcard masks.
- After an ACL is configured, it is linked to an interface using the **ip access-group** command in interface configuration mode.

Chapter 9: Summary (cont.)

- Remember the three Ps, 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.

Chapter 9: Summary (cont.)

- The **`access-class`** command configured in line configuration mode restricts incoming and outgoing connections between a particular VTY and the addresses in an access list.
- Like IPv4 named ACLs, IPv6 names are alphanumeric, case sensitive and must be unique. Unlike IPv4, there is no need for a standard or extended option.
- From global configuration mode, use the **`ipv6 access-list`** *`name`* command to create an IPv6 ACL. 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.