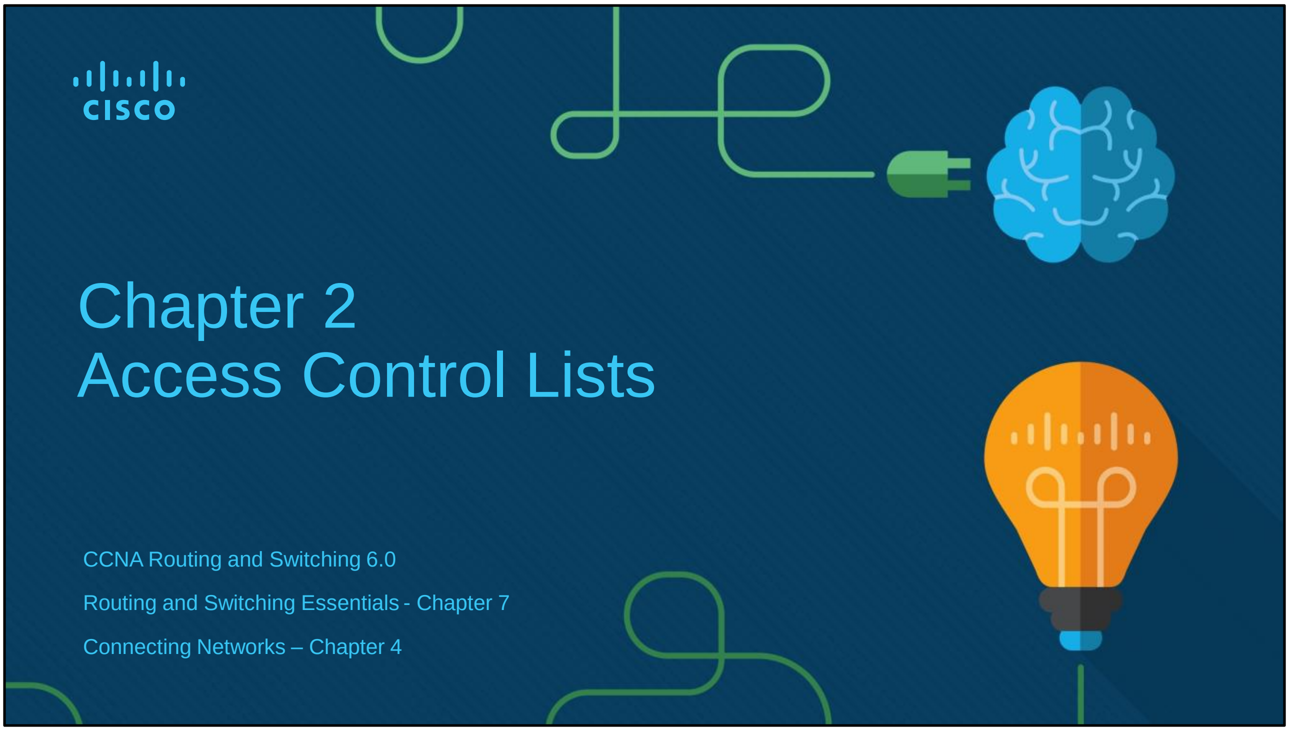
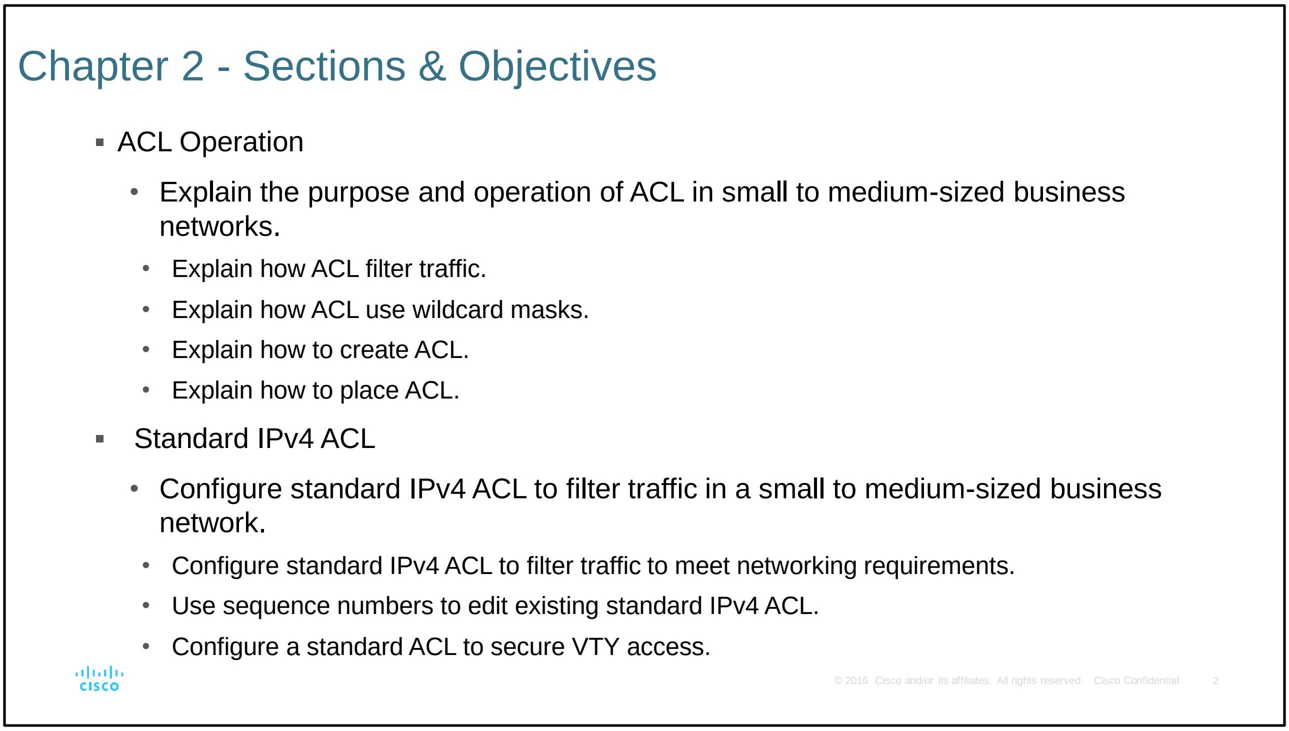


Chapter 2
Access Control Lists

CCNA Routing and Switching 6.0

Routing and Switching Essentials - Chapter 7

Connecting Networks – Chapter 4



Chapter 2 - Sections & Objectives

- ACL Operation
  - Explain the purpose and operation of ACL in small to medium-sized business networks.
    - Explain how ACL filter traffic.
    - Explain how ACL use wildcard masks.
    - Explain how to create ACL.
    - Explain how to place ACL.
- Standard IPv4 ACL
  - Configure standard IPv4 ACL to filter traffic in a small to medium-sized business network.
    - Configure standard IPv4 ACL to filter traffic to meet networking requirements.
    - Use sequence numbers to edit existing standard IPv4 ACL.
    - Configure a standard ACL to secure VTY access.

Access Control Lists

One of the most important skills a network administrator needs is mastery of access control lists (ACL). ACL provide security for a network.

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.

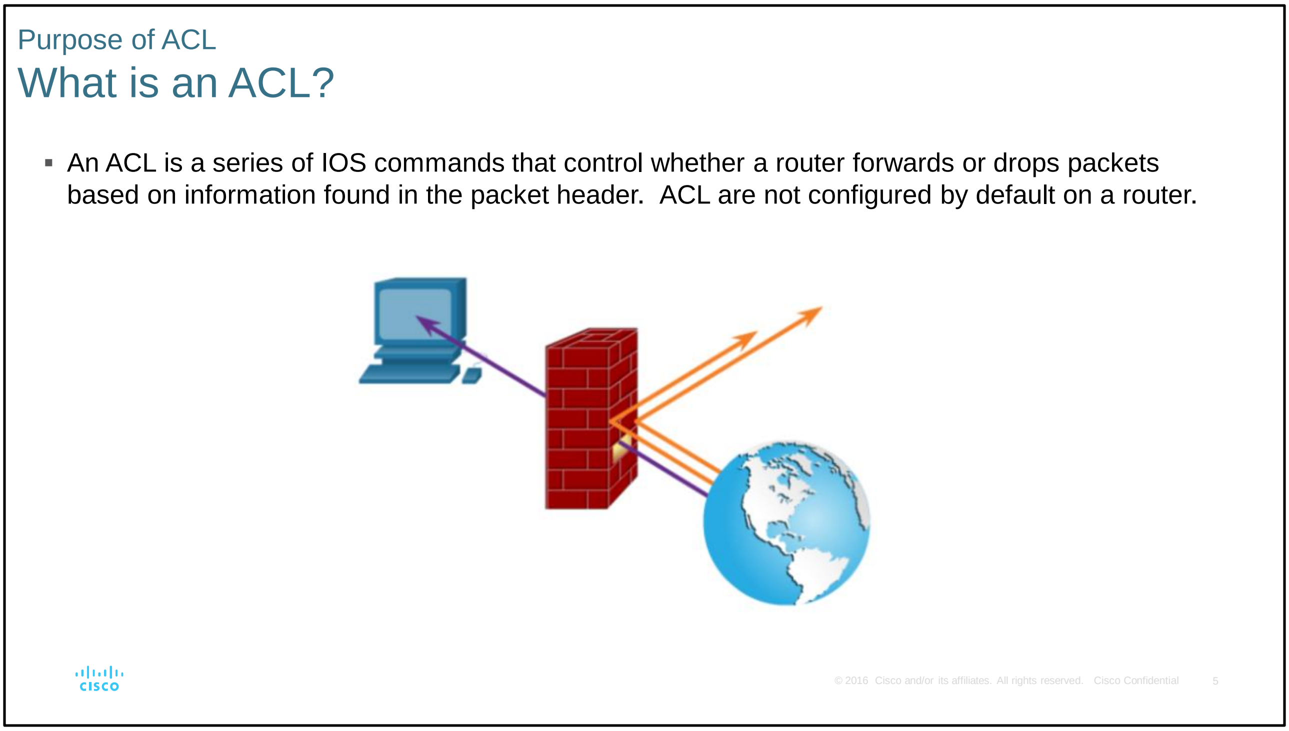
On a Cisco router, you can configure a simple firewall that provides basic traffic filtering capabilities using ACL. Administrators use ACL to stop traffic or permit only specified traffic on their networks.

This chapter explains how to configure and troubleshoot standard IPv4 ACL on a Cisco router as part of a security solution. Included are tips, considerations, recommendations, and general guidelines on how to use ACL. In addition, this chapter includes an opportunity to develop your mastery of ACL with a series of lessons, activities, and lab exercises.

Chapter 7 - Sections & Objectives (Cont.)

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

ACL Operation



Purpose of ACL

What is an ACL?

- An ACL is a series of IOS commands that control whether a router forwards or drops packets based on information found in the packet header. ACL are not configured by default on a router.

© 2016 Cisco and/or its affiliates. All rights reserved.   Cisco Confidential   5

What is an ACL?

An ACL is a series of IOS commands that control whether a router forwards or drops packets based on information found in the packet header. ACL are among the most commonly used features of Cisco IOS software.

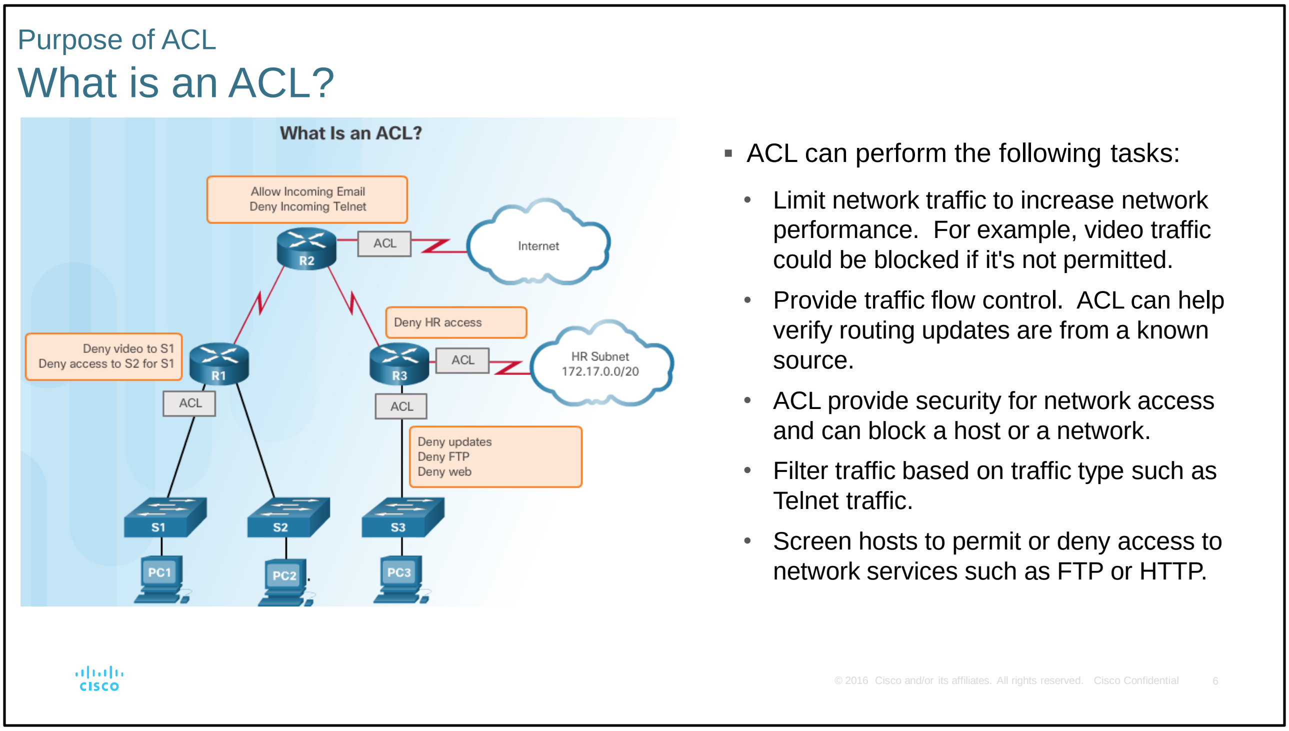
When configured, ACL perform the following tasks:

Limit network traffic to increase network performance. For example, if corporate policy does not allow video traffic on the network, ACL that block video traffic could be configured and applied. This would greatly reduce the network load and increase network performance.

Provide traffic flow control. ACL can restrict the delivery of routing updates to ensure that the updates are from a known source.

Provide a basic level of security for network access. ACL 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.



Purpose of ACL

What is an ACL?

- ACL can perform the following tasks:
  - Limit network traffic to increase network performance. For example, video traffic could be blocked if it's not permitted.
  - Provide traffic flow control. ACL can help verify routing updates are from a known source.
  - ACL provide security for network access and can block a host or a network.
  - Filter traffic based on traffic type such as Telnet traffic.
  - Screen hosts to permit or deny access to network services such as FTP or HTTP.

What is an ACL?

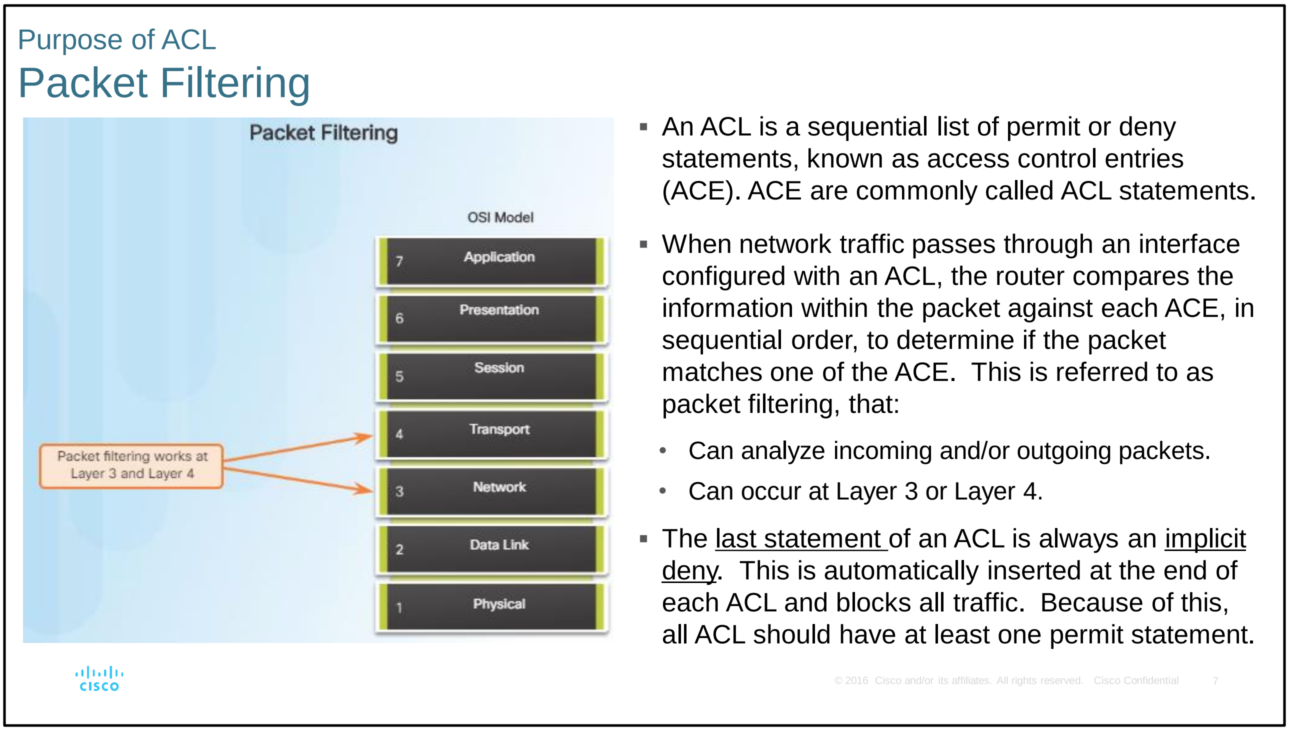
For example, an ACL can permit email traffic, but block all Telnet traffic.

Screen hosts to permit or deny access to network services. ACL can permit or deny a user to access file types, such as FTP or HTTP.

By default, a router does not have ACL 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.

In addition to either permitting or denying traffic, ACL can be used for selecting types of traffic to be analyzed, forwarded, or processed in other ways. For example, ACL can be used to classify traffic to enable priority processing. This capability is similar to having a VIP pass at a concert or sporting event. The VIP pass gives selected guests privileges not offered to general admission ticket holders, such as priority entry or being able to enter a restricted area.

The figure shows a sample topology with ACL applied.



Purpose of ACL

Packet Filtering

Packet Filtering
OSI Model
7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data Link
1 Physical
Packet filtering works at Layer 3 and Layer 4

- An ACL is a sequential list of permit or deny statements, known as access control entries (ACE). ACE are commonly called ACL statements.
- 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 ACE. This is referred to as packet filtering, that:
  - Can analyze incoming and/or outgoing packets.
  - Can occur at Layer 3 or Layer 4.
- The last statement of an ACL is always an implicit deny. This is automatically inserted at the end of each ACL and blocks all traffic. Because of this, all ACL should have at least one permit statement.

Packet Filtering

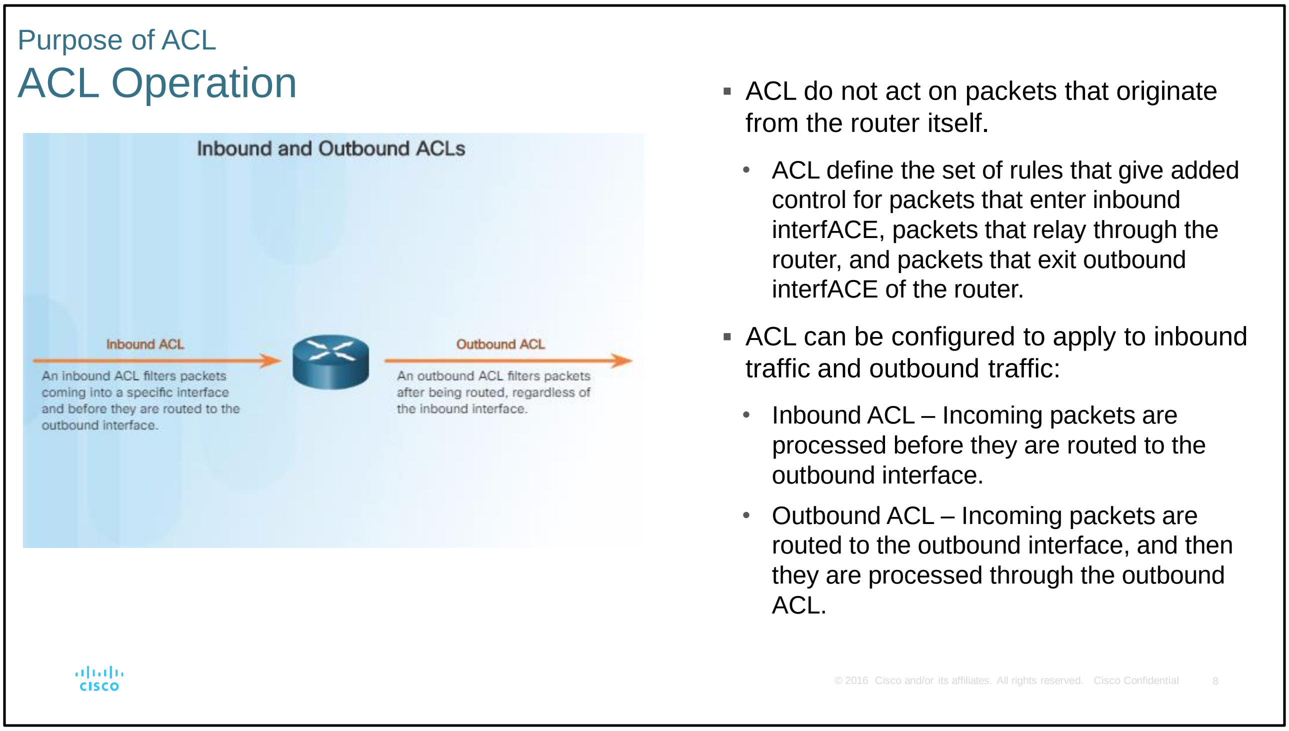
An ACL is a sequential list of permit or deny statements, known as access control entries (ACE). ACE are also commonly called ACL statements. When network traffic passes through an interface configured with an ACL, the router compares the information within the packet against each ACE, in sequential order, to determine if the packet matches one of the ACE. This process is called packet filtering.

Packet filtering controls access to a network by analyzing the incoming and outgoing packets and forwarding them or discarding them based on given criteria. Packet filtering can occur at Layer 3 or Layer 4, as shown in the figure. Standard ACL only filter at Layer 3. Extended ACL filter at Layer 3 and Layer 4.

Note: Extended ACL are beyond the scope of this course.

The source IPv4 address is the filtering criteria set in each ACE of a standard IPv4 ACL. A router configured with a standard IPv4 ACL extracts the source IPv4 address from the packet header. The router starts at the top of the ACL and compares the address to each ACE sequentially. When a match is made, the router carries out the instruction, either permitting or denying the packet. After a match is made, the remaining ACE in the ACL, if any, are not analyzed. If the source IPv4 address does not match any ACE in the ACL, the packet is discarded.

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.



Purpose of ACL

ACL Operation

Inbound and Outbound ACLs
Inbound ACL
An inbound ACL filters packets coming into a specific interface and before they are routed to the outbound interface.
Outbound ACL
An outbound ACL filters packets after being routed, regardless of the inbound interface.

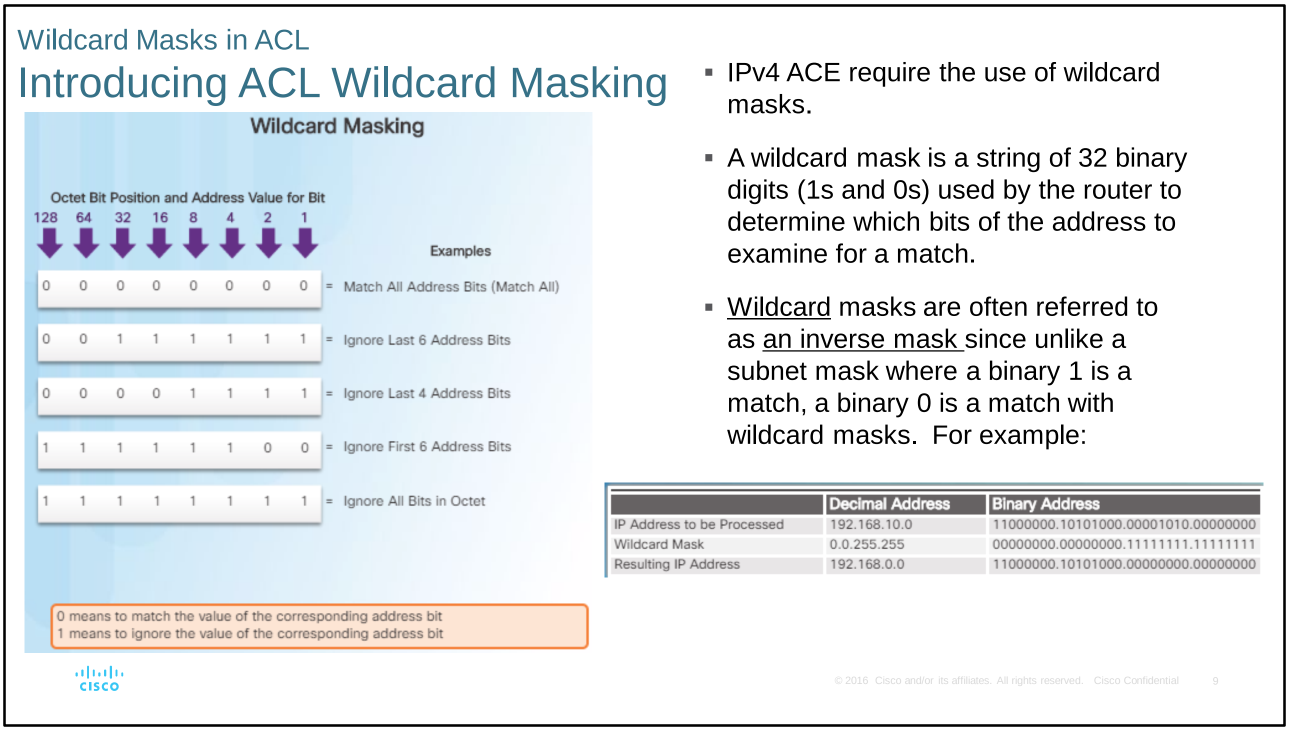
- ACL do not act on packets that originate from the router itself.
  - ACL define the set of rules that give added control for packets that enter inbound interfACE, packets that relay through the router, and packets that exit outbound interfACE of the router.
- ACL can be configured to apply to inbound traffic and outbound traffic:
  - Inbound ACL – Incoming packets are processed before they are routed to the outbound interface.
  - Outbound ACL – Incoming packets are routed to the outbound interface, and then they are processed through the outbound ACL.

ACL Operation

ACL define the set of rules that give added control for packets that enter inbound interfACE, packets that relay through the router, and packets that exit outbound interfACE of the router. ACL do not act on packets that originate from the router itself. ACL can be configured to apply to inbound traffic and outbound traffic as shown in the figure.

Inbound ACL - 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 ACL, it is then processed for routing. Inbound ACL are best used to filter packets when the network attached to an inbound interface is the only source of packets that need to be examined.

Outbound ACL - Incoming packets are routed to the outbound interface, and then they are processed through the outbound ACL. Outbound ACL are best used when the same filter will be applied to packets coming from multiple inbound interfACE before exiting the same outbound interface.



Wildcard Masks in ACL

Introducing ACL Wildcard Masking

Wildcard Masking

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

0 means to match the value of the corresponding address bit
1 means to ignore the value of the corresponding address bit

- IPv4 ACE require the use of wildcard masks.
- A wildcard mask is a string of 32 binary digits (1s and 0s) used by the router to determine which bits of the address to examine for a match.
- Wildcard masks are often referred to as an inverse mask since unlike a subnet mask where a binary 1 is a match, a binary 0 is a match with wildcard masks. For example:

	Decimal Address	Binary Address
IP Address to be Processed	192.168.10.0	11000000.10101000.00001010.00000000
Wildcard Mask	0.0.255.255	00000000.00000000.11111111.11111111
Resulting IP Address	192.168.0.0	11000000.10101000.00000000.00000000

ntroducing ACL Wildcard Masking

Wildcard Masking

IPv4 ACE 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.

As with subnet masks, the numbers 1 and 0 in the wildcard mask identify how to treat the corresponding IPv4 address bits. However, in a wildcard mask, these bits are used for different purposes and follow different rules.

Subnet masks use binary 1s and 0s to identify the network, subnet, and host portion of an IPv4 address. Wildcard masks use binary 1s and 0s to filter individual IPv4 addresses or groups of IPv4 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.

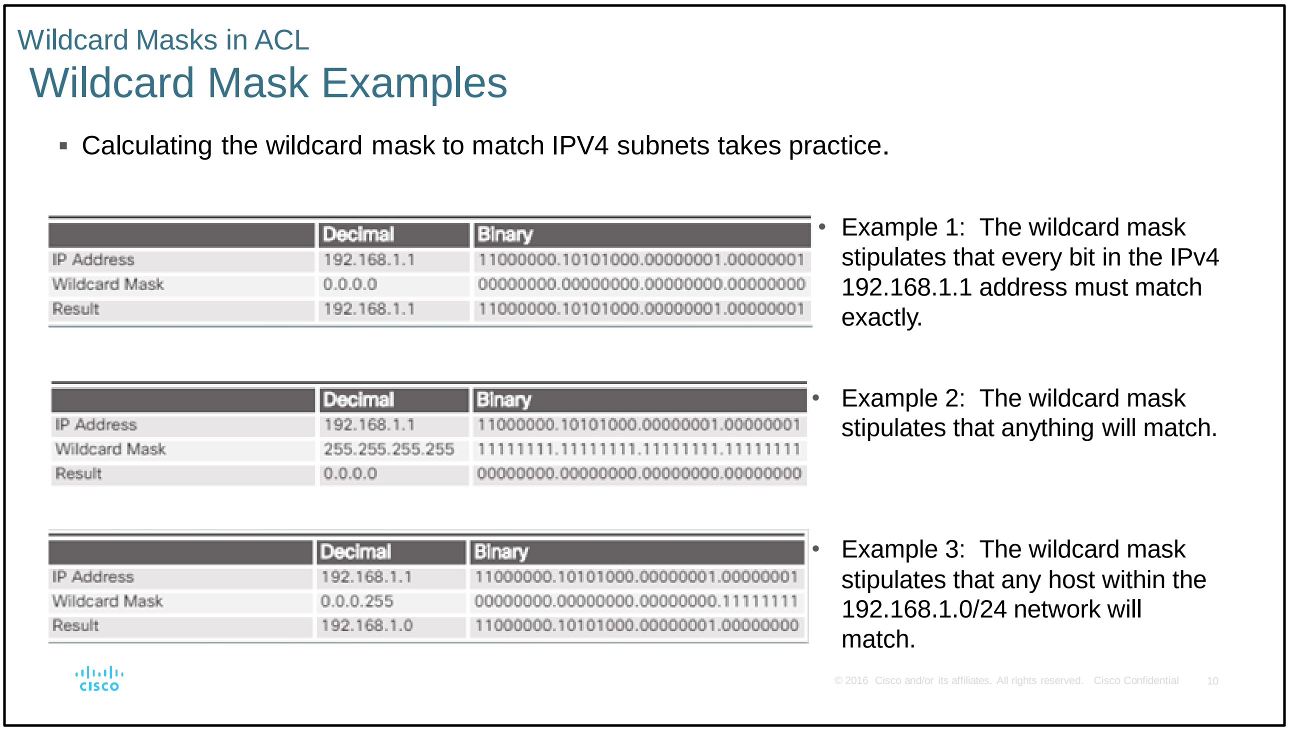
Wildcard mask bit 1 - Ignore the corresponding bit value in the address.

Figure 1 shows how different wildcard masks filter IPv4 addresses. In the example, remember that binary 0 signifies a bit that must match, and binary 1 signifies a bit that can be ignored.

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.

Using a Wildcard Mask

The table in Figure 2 shows the results of applying a 0.0.255.255 wildcard mask to a 32-bit IPv4 address. Remember that a binary 0 indicates a value that is matched.
Note: Unlike IPv4 ACL, IPv6 ACL 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. IPv6 ACL are beyond the scope of this course.



Wildcard Masks in ACL

Wildcard Mask Examples

- Calculating the wildcard mask to match IPV4 subnets takes practice.

	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 1: The wildcard mask stipulates that every bit in the IPv4 192.168.1.1 address must match exactly.

	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 2: The wildcard mask stipulates that anything will match.

	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

- Example 3: The wildcard mask stipulates that any host within the 192.168.1.0/24 network will match.

Wildcard Mask Examples

Wildcard Masks to Match IPv4 Subnets

Calculating the wildcard mask can take some practice. Figure 1 provides three examples of wildcard masks.

In the first example the wildcard mask stipulates that every bit in the IPv4 192.168.1.1 must match exactly.

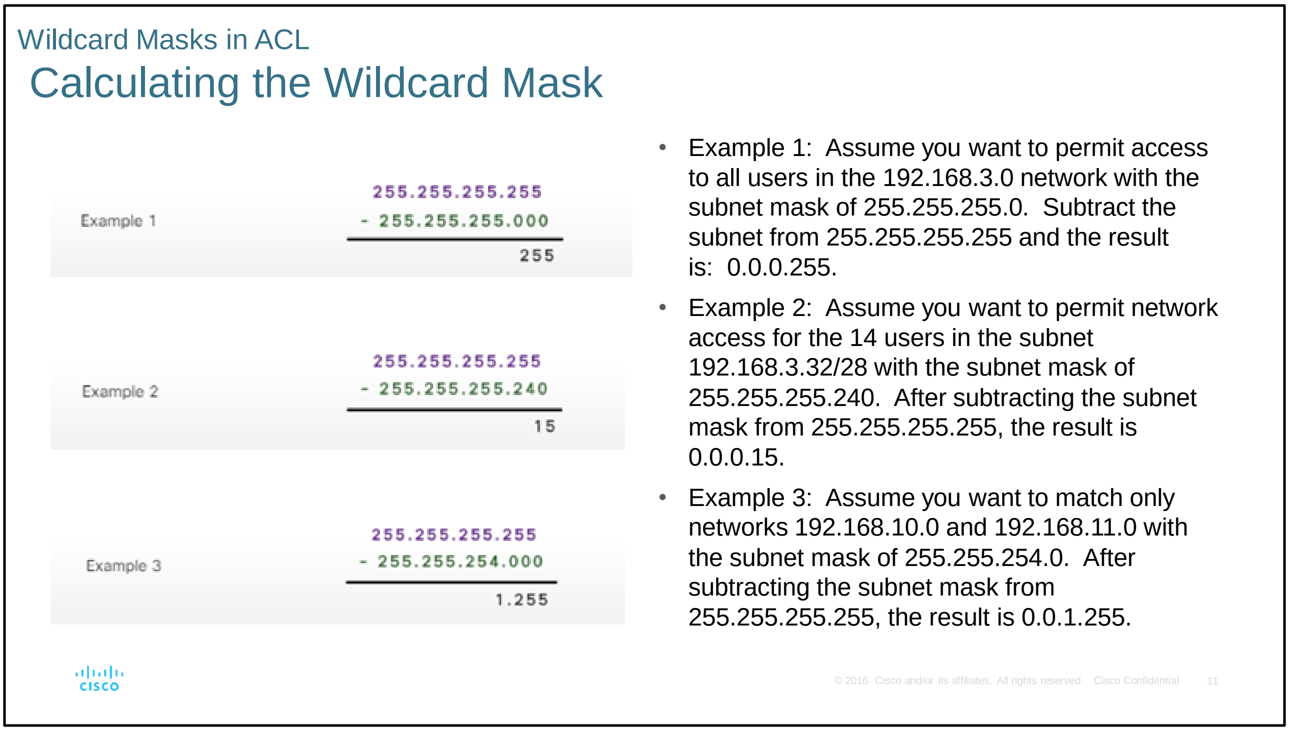
In the second example, the wildcard mask stipulates that anything will match.

In the third example, the wildcard mask stipulates that any host within the 192.168.1.0/24 network will match.

Wildcard Masks to Match Ranges

The two examples in Figure 2 are more complex. In example 1, the first two octets and first four bits of the third octet must match exactly. The last four bits in the third octet and the last octet can be any valid number. This results in a mask that checks for the range of networks 192.168.16.0 to 192.168.31.0.

Example 2 shows a wildcard mask that matches the first two octets, and the least significant bit in the third octet. The last octet and the first seven bits in the third octet can be any valid number. The result is a mask that would permit or deny all hosts from odd subnets from the 192.168.0.0 major network.



Wildcard Masks in ACL

Calculating the Wildcard Mask

Example 1

```
  255.255.255.255
- 255.255.255.000
-----------------
              255
```

Example 2

```
  255.255.255.255
- 255.255.255.240
-----------------
               15
```

Example 3

```
  255.255.255.255
- 255.255.254.000
-----------------
            1.255
```

- Example 1: Assume you want to permit access to all users in the 192.168.3.0 network with the subnet mask of 255.255.255.0. Subtract the subnet from 255.255.255.255 and the result is: 0.0.0.255.
- Example 2: Assume you want to permit network access for the 14 users in the subnet 192.168.3.32/28 with the subnet mask of 255.255.255.240. After subtracting the subnet mask from 255.255.255.255, the result is 0.0.0.15.
- Example 3: Assume you want to match only networks 192.168.10.0 and 192.168.11.0 with the subnet mask of 255.255.254.0. After subtracting the subnet mask from 255.255.255.255, the result is 0.0.1.255.

Calculating the Wildcard Mask

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

Wildcard Mask Calculation: Example 1

In the first example in the figure, assume you wanted to permit access to all users in the 192.168.3.0 network. Because the subnet mask is 255.255.255.0, you could take the 255.255.255.255 and subtract the subnet mask 255.255.255.0. The solution produces the wildcard mask 0.0.0.255.

Wildcard Mask Calculation: Example 2

In the second example in the figure, assume you wanted to permit network access for the 14 users in the subnet 192.168.3.32/28. The subnet mask for the IPv4 subnet is 255.255.255.240, therefore take 255.255.255.255 and subtract the subnet mask 255.255.255.240. The solution this time produces the wildcard mask 0.0.0.15.

Wildcard Mask Calculation: Example 3

In the third example in the figure, assume you wanted to match only networks 192.168.10.0 and 192.168.11.0. Again, you take the 255.255.255.255 and subtract the regular subnet mask which in this case would be 255.255.255.0. The result is 0.0.0.255.

You could accomplish the same result with statements like the two shown below:

R1(config)# **access-list 10 permit 192.168.10.0 0.0.0.255**

R1(config)# **access-list 10 permit 192.168.11.0 0.0.0.255**

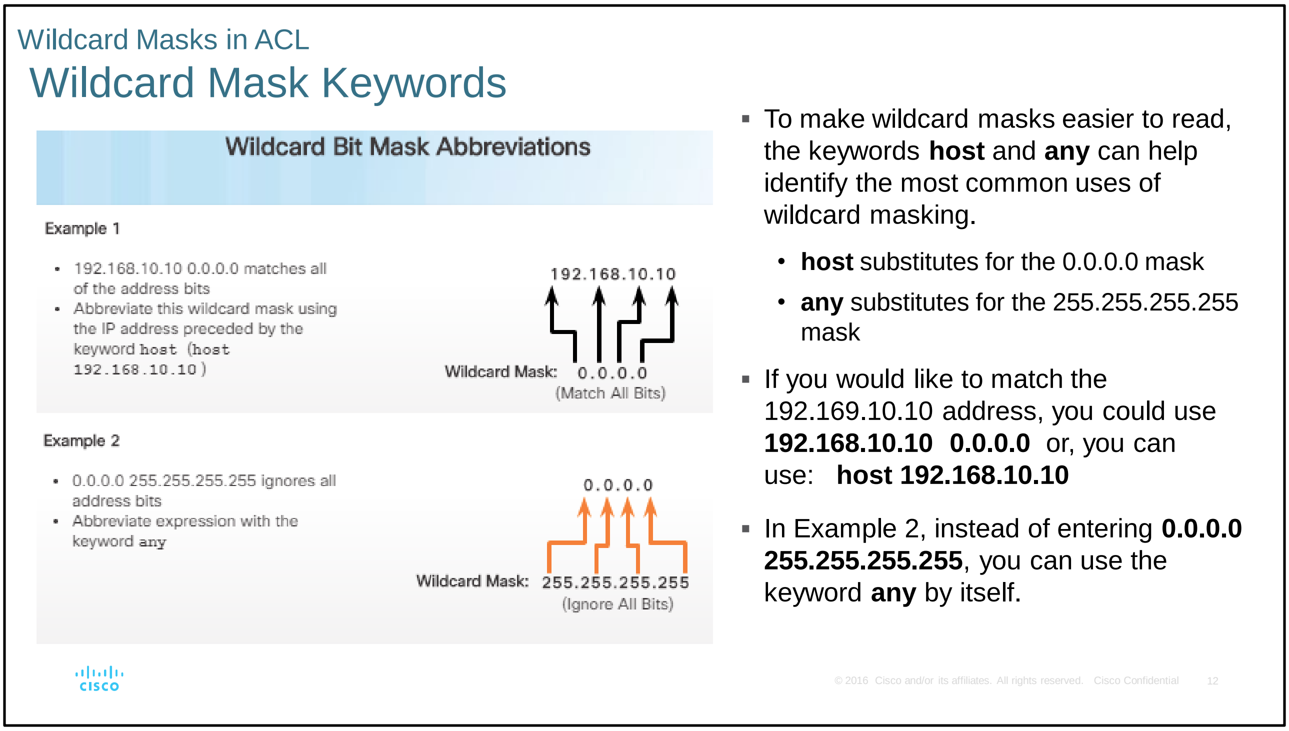
But It is far more efficient to configure the wildcard mask in the following way:

```
R1(config)# access-list 10 permit 192.168.10.0 0.0.1.255
```

The wildcard mask 0.0.1.255 would restrict the permitted subnets to 192.168.10.0 and 192.168.11.0

Consider an example in which you need to match networks in the range between 192.168.16.0/24 to 192.168.31.0/24. These networks would summarize to 192.168.16.0/20. In this case, 0.0.15.255 is the correct wildcard mask to configure one efficient ACL statement, as shown below:

```
R1(config)# access-list 10 permit 192.168.16.0 0.0.15.255
```



Wildcard Masks in ACL

Wildcard Mask Keywords

Wildcard Bit Mask Abbreviations

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`)

192.168.10.10
Wildcard Mask: 0.0.0.0
(Match All Bits)

Example 2

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

0.0.0.0
Wildcard Mask: 255.255.255.255
(Ignore All Bits)

- To make wildcard masks easier to read, the keywords **host** and **any** can help identify the most common uses of wildcard masking.
  - **host** substitutes for the 0.0.0.0 mask
  - **any** substitutes for the 255.255.255.255 mask
- If you would like to match the 192.169.10.10 address, you could use **192.168.10.10 0.0.0.0** or, you can use: **host 192.168.10.10**
- In Example 2, instead of entering **0.0.0.0 255.255.255.255**, you can use the keyword **any** by itself.

Wildcard Mask Keywords

Working with decimal representations of binary wildcard mask bits can be tedious. To simplify this task, the keywords **host** and **any** help identify the most common uses of wildcard masking. These keywords eliminate entering wildcard masks when identifying a specific host or an entire network. These keywords also make it easier to read an ACL by providing visual clues as to the source or destination of the criteria.

The **host** keyword substitutes for the 0.0.0.0 mask. This mask states that all IPv4 address bits must match to filter just one host address.

The **any** option substitutes for the IPv4 address and 255.255.255.255 mask. This mask says to ignore the entire IPv4 address or to accept any addresses.

Example 1: Wildcard Masking Process with a Single IPv4 Address

In Example 1 in the figure, instead of entering **192.168.10.10 0.0.0.0**, you can use **host 192.168.10.10**.

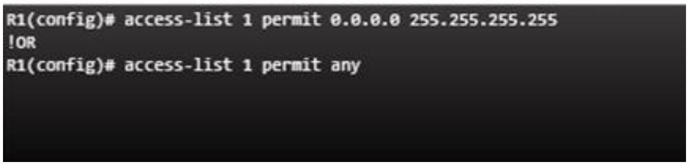
Example 2: Wildcard Masking Process with a Match Any IPv4 Address

In Example 2 in the figure, instead of entering **0.0.0.0 255.255.255.255**, you can use the keyword **any** by itself.

Wildcard Masks in ACL

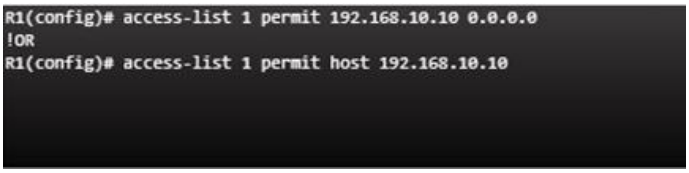
Wildcard Mask Keyword Examples

- Example 1 in the figure demonstrates how to use the **any** keyword to substitute the IPv4 address 0.0.0.0 with a wildcard mask of 255.255.255.255.



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

- Example 2 demonstrates how to use the **host** keyword to substitute for the wildcard mask when identifying a single host.



```
R1(config)# access-list 1 permit 192.168.10.10 0.0.0.0
!OR
R1(config)# access-list 1 permit host 192.168.10.10
```

Wildcard Mask Keyword Examples

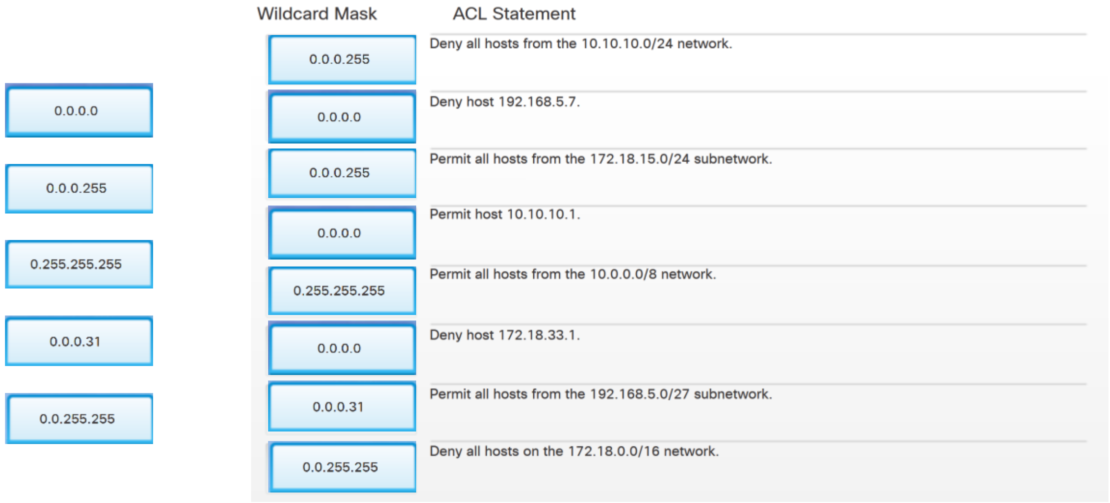
Example 1 in the figure shows how to use the **any** keyword to substitute the IPv4 address 0.0.0.0 with a wildcard mask of 255.255.255.255.

Example 2 shows how to use the **host** keyword to substitute for the wildcard mask when identifying a single host.

Note: The syntax for configuring standard IPv4 ACL is covered later in this chapter.

Guidelines for ACL Creation

Activity – Determine the correct wildcard mask



0.0.0.0

0.0.0.255

0.255.255.255

0.0.0.31

0.0.255.255

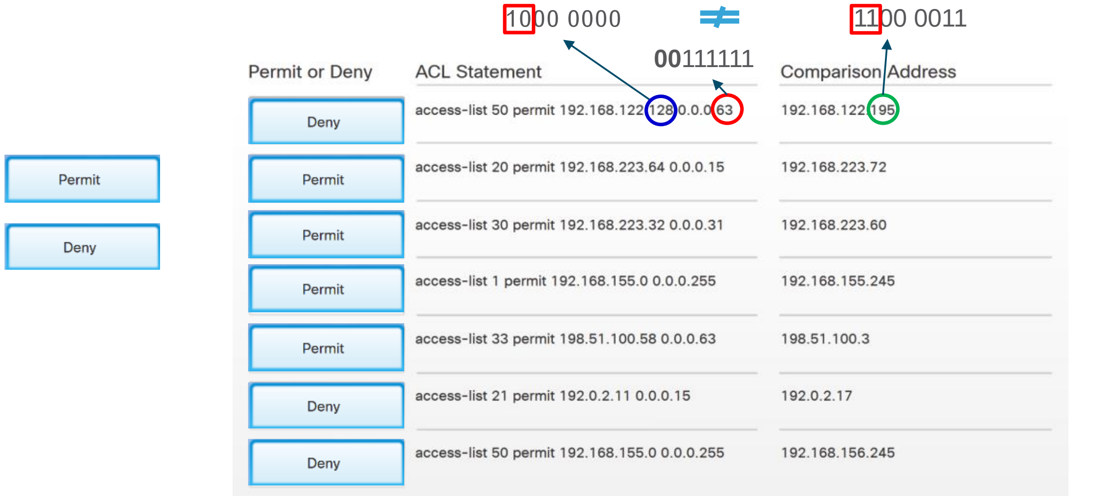
Wildcard Mask	ACL Statement
0.0.0.255	Deny all hosts from the 10.10.10.0/24 network.
0.0.0.0	Deny host 192.168.5.7.
0.0.0.255	Permit all hosts from the 172.18.15.0/24 subnetwork.
0.0.0.0	Permit host 10.10.10.1.
0.255.255.255	Permit all hosts from the 10.0.0.0/8 network.
0.0.0.0	Deny host 172.18.33.1.
0.0.0.31	Permit all hosts from the 192.168.5.0/27 subnetwork.
0.0.255.255	Deny all hosts on the 172.18.0.0/16 network.

7.1 – ACL Operation
7.1.3 – Guidelines for ACL Creation
7.1.3.1 - General guidelines for Creating ACL

Guidelines for ACL Creation

Activity – Determine the Permit or Deny

128-64-32-16-8-4-2-1



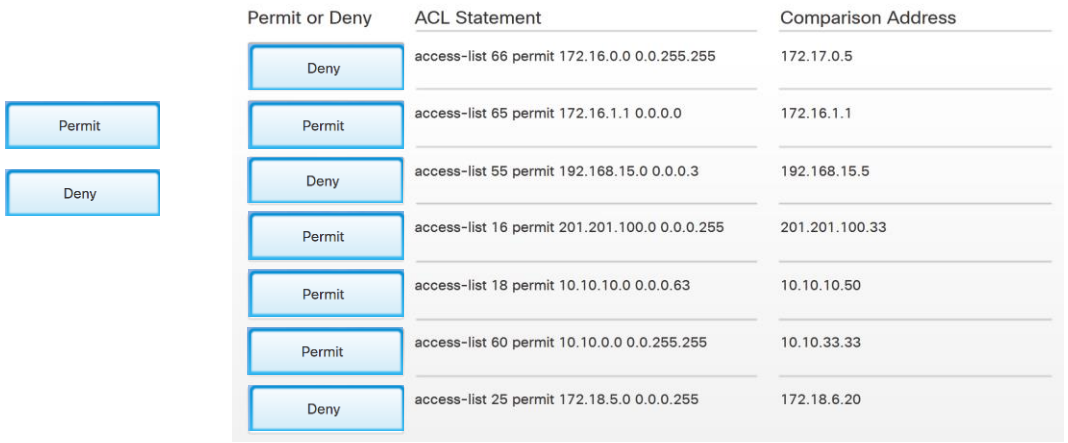
Permit or Deny	ACL Statement	Comparison Address
Deny	access-list 50 permit 192.168.122.128 0.0.0.63	192.168.122.195
Permit	access-list 20 permit 192.168.223.64 0.0.0.15	192.168.223.72
Permit	access-list 30 permit 192.168.223.32 0.0.0.31	192.168.223.60
Permit	access-list 1 permit 192.168.155.0 0.0.0.255	192.168.155.245
Permit	access-list 33 permit 198.51.100.58 0.0.0.63	198.51.100.3
Deny	access-list 21 permit 192.0.2.11 0.0.0.15	192.0.2.17
Deny	access-list 50 permit 192.168.155.0 0.0.0.255	192.168.156.245

7.1 – ACL Operation
7.1.3 – Guidelines for ACL Creation
7.1.3.1 - General guidelines for Creating ACL

Guidelines for ACL Creation

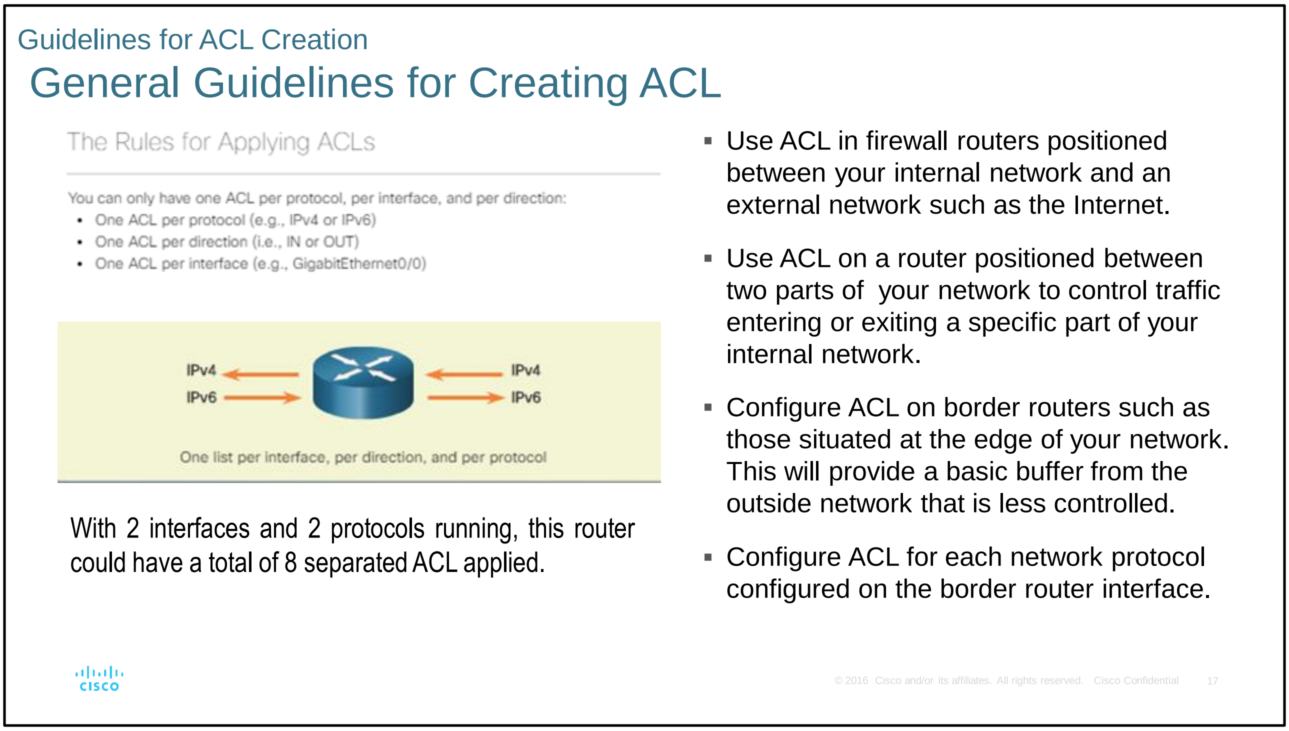
Activity – Determine the Permit or Deny

128-64-32-16-8-4-2-1



Permit or Deny	ACL Statement	Comparison Address
Deny	access-list 66 permit 172.16.0.0 0.0.255.255	172.17.0.5
Permit	access-list 65 permit 172.16.1.1 0.0.0.0	172.16.1.1
Deny	access-list 55 permit 192.168.15.0 0.0.0.3	192.168.15.5
Permit	access-list 16 permit 201.201.100.0 0.0.0.255	201.201.100.33
Permit	access-list 18 permit 10.10.10.0 0.0.0.63	10.10.10.50
Permit	access-list 60 permit 10.10.0.0 0.0.255.255	10.10.33.33
Deny	access-list 25 permit 172.18.5.0 0.0.0.255	172.18.6.20

7.1 – ACL Operation
7.1.3 – Guidelines for ACL Creation
7.1.3.1 - General guidelines for Creating ACL



Guidelines for ACL Creation

General Guidelines for Creating ACL

The Rules for Applying ACLs

You can only have one ACL per protocol, per interface, and per direction:

- One ACL per protocol (e.g., IPv4 or IPv6)
- One ACL per direction (i.e., IN or OUT)
- One ACL per interface (e.g., GigabitEthernet0/0)

IPv4
IPv6
IPv4
IPv6

One list per interface, per direction, and per protocol

With 2 interfaces and 2 protocols running, this router could have a total of 8 separated ACL applied.

- Use ACL in firewall routers positioned between your internal network and an external network such as the Internet.
- Use ACL on a router positioned between two parts of your network to control traffic entering or exiting a specific part of your internal network.
- Configure ACL on border routers such as those situated at the edge of your network. This will provide a basic buffer from the outside network that is less controlled.
- Configure ACL for each network protocol configured on the border router interface.

General Guidelines for Creating ACL

Writing ACL 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 interface configured for IPv4 and IPv6. If we needed ACL for both protocols, on both interfACE and in both directions, this would require eight separate ACL. Each interface would have four ACL; two ACL for IPv4 and two ACL for IPv6. For each protocol, one ACL is for inbound traffic and one for outbound traffic.

Note: ACL do not have to be configured in both directions. The number of ACL and their direction applied to the interface will depend on the requirements being implemented.

Here are some guidelines for using ACL:

Use ACL in firewall routers positioned between your internal network and an external network such as the Internet.

Use ACL on a router positioned between two parts of your network to control traffic entering or exiting a specific part of your internal network.

Configure ACL on border routers, that is, routers situated at the edges of your networks. This provides a very basic buffer from the outside network, or between a less controlled area of your own network and a more sensitive area of your network.

Configure ACL for each network protocol configured on the border router interfACE.

Rules for Applying ACL

You can configure one ACL per protocol, per direction, per interface:

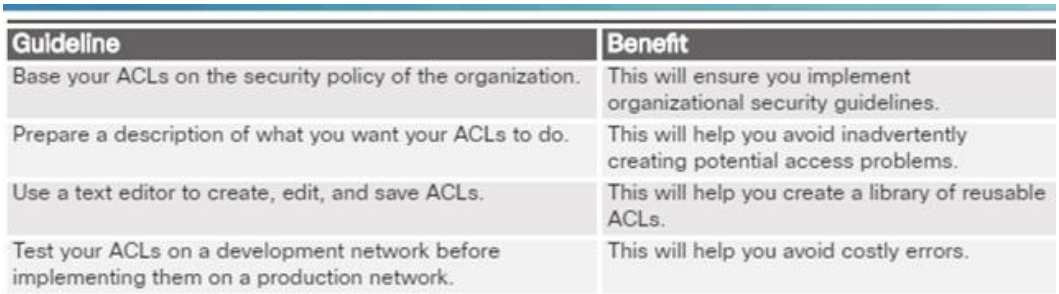
One ACL per protocol - To control traffic flow on an interface, an ACL must be defined

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

Guidelines for ACL Creation

ACL Best Practices

- Using ACL requires significant attention to detail. Mistakes can be very costly in terms of downtime, troubleshooting efforts, and poor network performance.



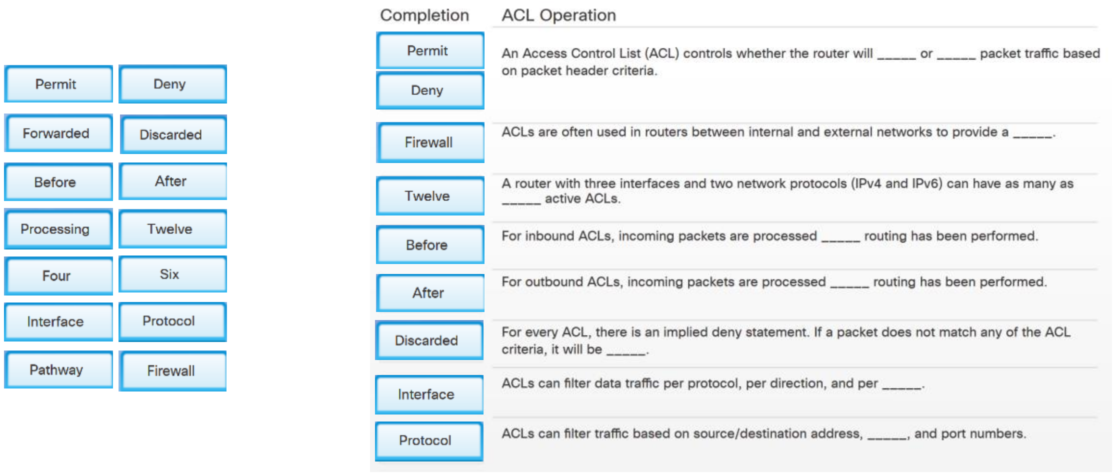
Guideline	Benefit
Base your ACLs on the security policy of the organization.	This will ensure you implement organizational security guidelines.
Prepare a description of what you want your ACLs to do.	This will help you avoid inadvertently creating potential access problems.
Use a text editor to create, 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.

ACL Best Practices

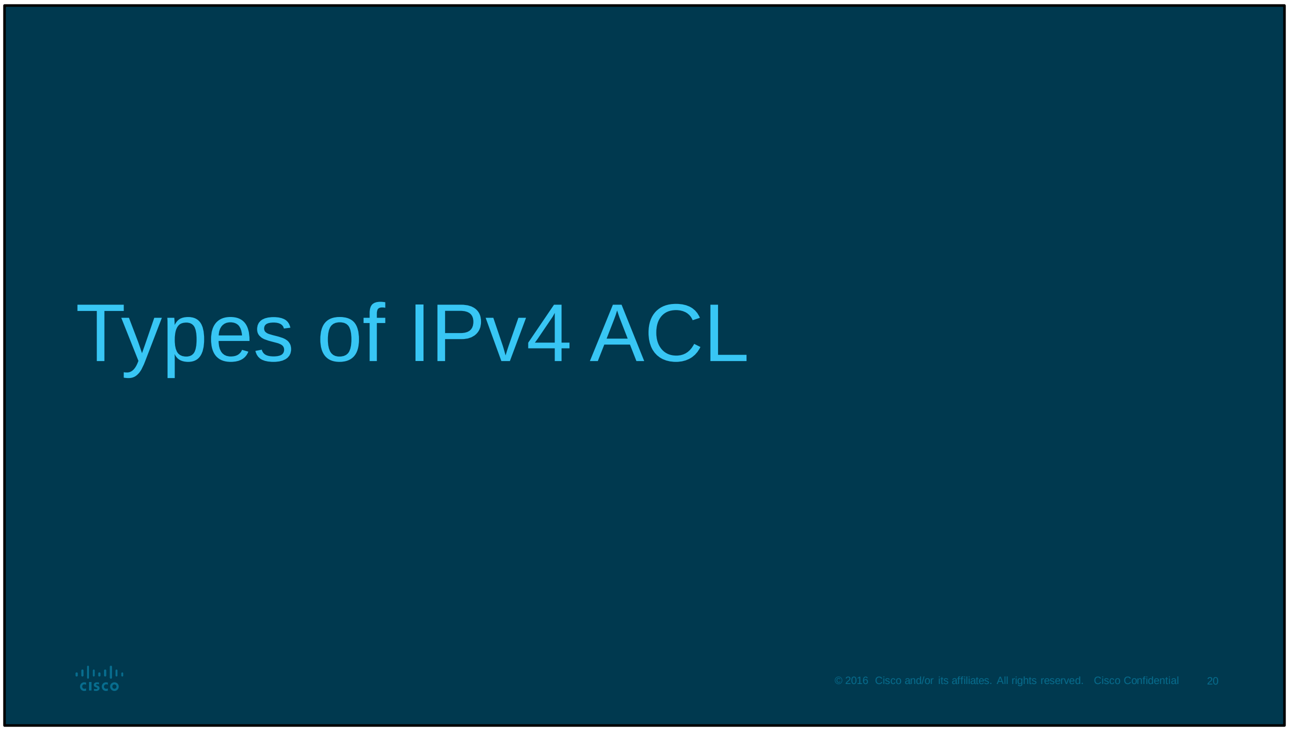
Using ACL requires attention to detail and great care. Mistakes can be costly in terms of downtime, troubleshooting efforts, and poor network service. Before configuring an ACL, basic planning is required. The figure presents guidelines that form the basis of an ACL best practices list.

Guidelines for ACL Creation

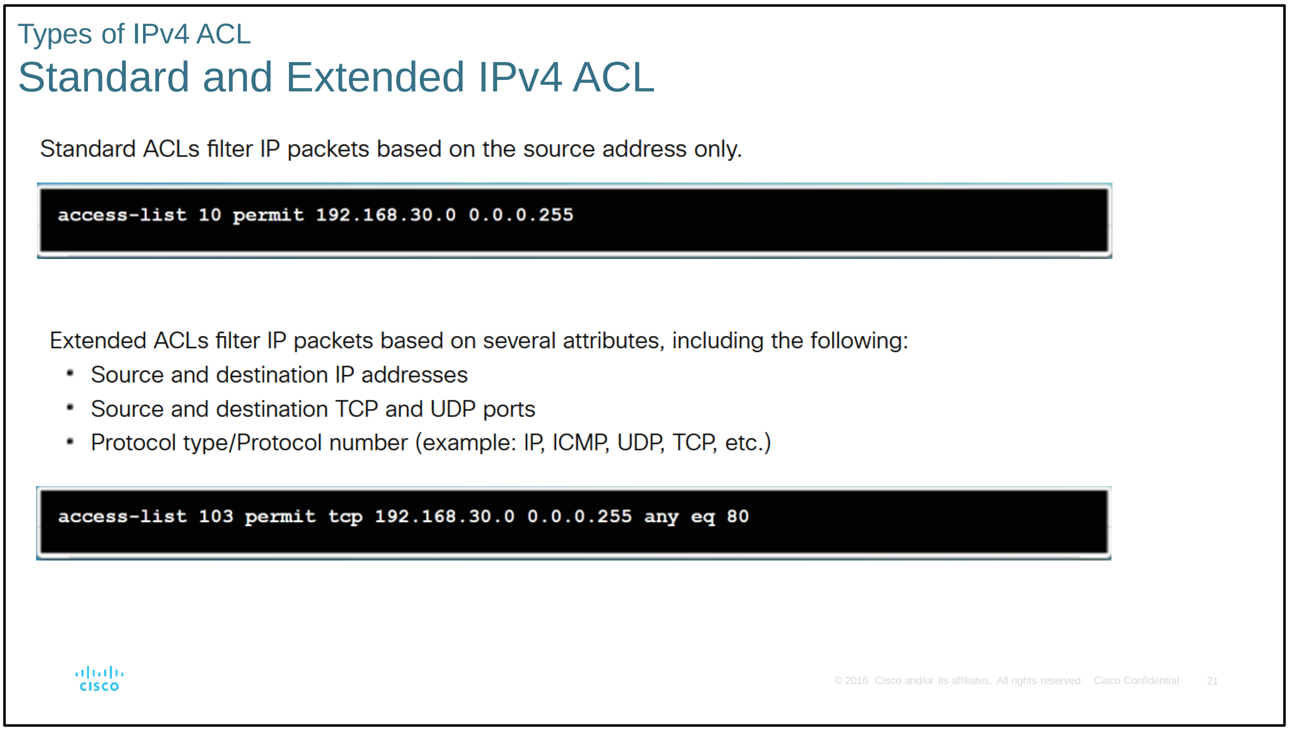
Activity – ACL Operation



Completion	ACL Operation
Permit Deny	An Access Control List (ACL) controls whether the router will _____ or _____ packet traffic based on packet header criteria.
Firewall	ACLs are often used in routers between internal and external networks to provide a _____.
Twelve	A router with three interfaces and two network protocols (IPv4 and IPv6) can have as many as _____ active ACLs.
Before	For inbound ACLs, incoming packets are processed _____ routing has been performed.
After	For outbound ACLs, incoming packets are processed _____ routing has been performed.
Discarded	For every ACL, there is an implied deny statement. If a packet does not match any of the ACL criteria, it will be _____.
Interface	ACLs can filter data traffic per protocol, per direction, and per _____.
Protocol	ACLs can filter traffic based on source/destination address, _____, and port numbers.



Types of IPv4 ACL



Types of IPv4 ACL

Standard and Extended IPv4 ACL

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

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

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, ICMP, UDP, TCP, etc.)

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

Standard and Extended IPv4 ACL

The two types of Cisco IPv4 ACL are standard and extended.

Standard ACL 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 in 1st figure allows all traffic from the 192.168.30.0/24 network. Because of the implied "deny any" at the end, all traffic except for traffic coming from the 192.168.30.0/24 network is blocked with this ACL. Standard ACL are created in global configuration mode.

Extended ACL 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

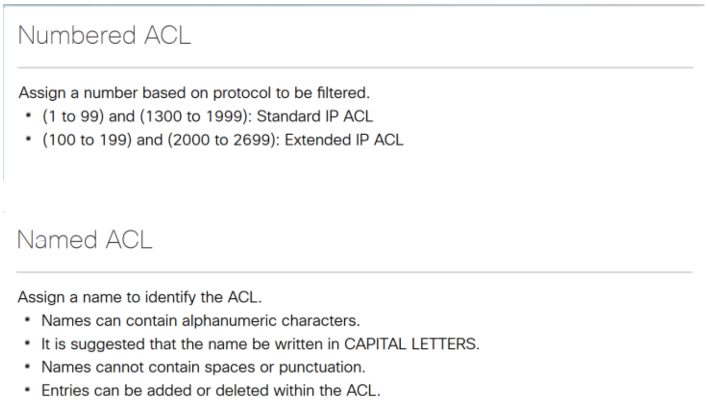
Optional protocol type information for finer control

In 2nd figure 2, 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 ACL are created in global configuration mode.

Note: ACL command syntax is discussed in more detail later in this chapter.

Types of IPv4 ACL

Numbered and Named IPv4 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.

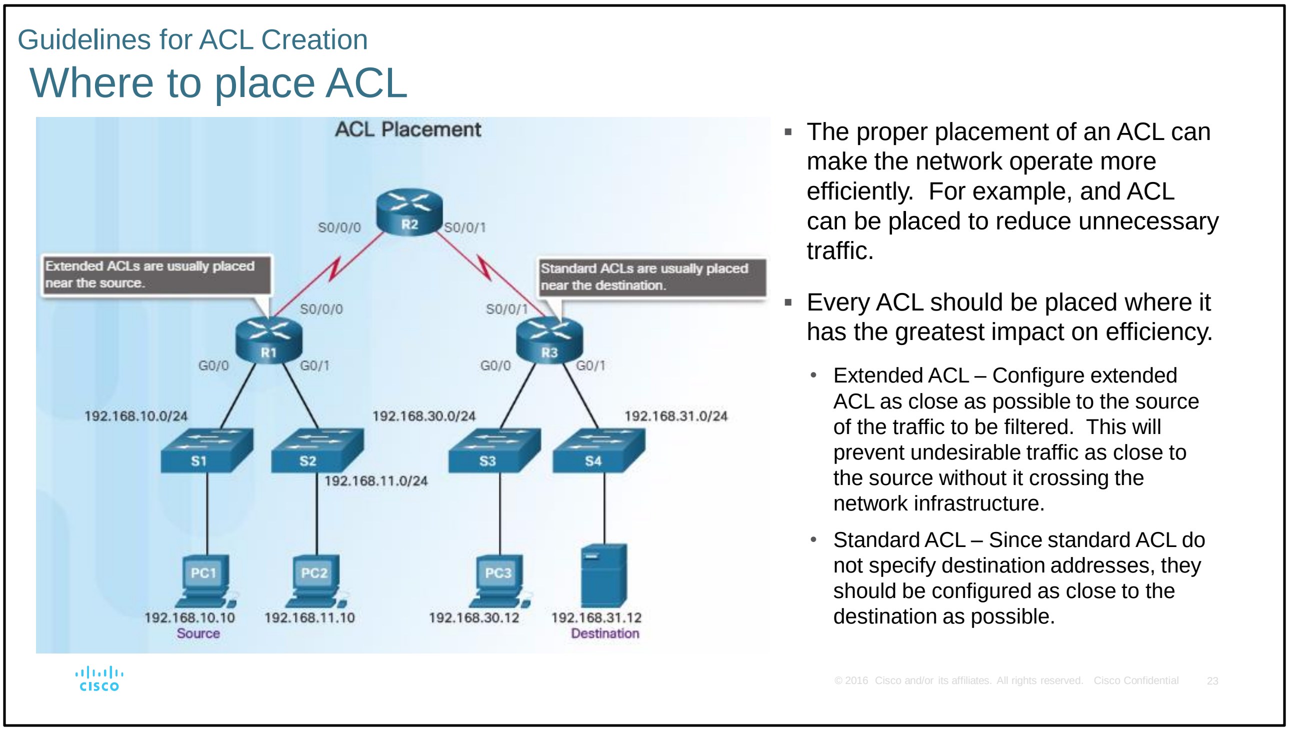
- Names can contain alphanumeric characters.
- It is suggested that the name be written in CAPITAL LETTERS.
- Names cannot contain spaces or punctuation.
- Entries can be added or deleted within the ACL.

- Standard and extended ACL can be created using either a number or a name to identify the ACL and its list of statements.
- Using numbered ACL 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. For this reason, a name can be used to identify a Cisco ACL.

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

Using numbered ACL 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. For this reason, a name can be used to identify a Cisco ACL.

The figure summarizes the rules to follow to designate numbered ACL and named ACL.



Guidelines for ACL Creation

Where to place ACL

- The proper placement of an ACL can make the network operate more efficiently. For example, and ACL can be placed to reduce unnecessary traffic.
- Every ACL should be placed where it has the greatest impact on efficiency.
  - Extended ACL – Configure extended ACL as close as possible to the source of the traffic to be filtered. This will prevent undesirable traffic as close to the source without it crossing the network infrastructure.
  - Standard ACL – Since standard ACL do not specify destination addresses, they should be configured as close to the destination as possible.

23

Where to Place ACL

Every ACL should be placed where it has the greatest impact on efficiency. As shown in the figure, the basic rules are:

Extended ACL - Locate extended ACL 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 ACL - Because standard ACL do not specify destination addresses, place them as close to the destination as possible. If a standard ACL was placed at the source of the traffic, the "permit" or "deny" will occur based on the given source address no matter where the traffic is destined.

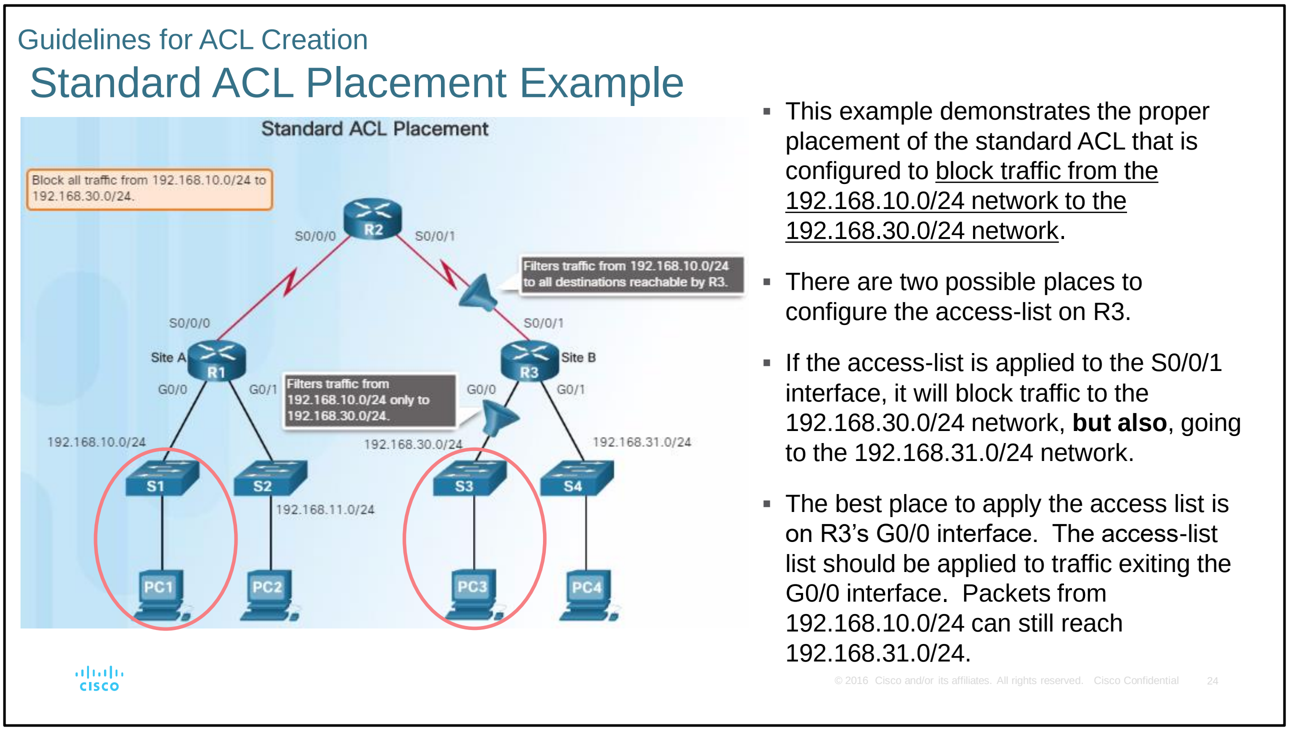
Placement of the ACL and therefore, the type of ACL used, may also depend a variety of factors:

The extent of the network administrator’s control - Placement of the ACL can depend on whether or not the network administrator has control of both the source and destination networks.

Bandwidth of the networks involved - Filtering unwanted traffic at the source prevents transmission of the traffic before it consumes bandwidth on the path to a destination. This is especially important in low bandwidth networks.

Ease of configuration - If a network administrator wants to deny traffic coming from several networks, one option is to use a single standard ACL on the router closest to the destination. The disadvantage is that traffic from these networks will use bandwidth unnecessarily. An extended ACL could be used on each router where the traffic originated. This will save bandwidth by filtering the traffic at the source, but requires creating extended ACL on multiple routers.

Note: For CCNA certification, the general rule is that extended ACL are placed as close as possible to the source and standard ACL are placed as close as possible to the destination.



Guidelines for ACL Creation

Standard ACL Placement Example

Standard ACL Placement
Block all traffic from 192.168.10.0/24 to 192.168.30.0/24.
R2
S0/0/0
S0/0/1
Filters traffic from 192.168.10.0/24 to all destinations reachable by R3.
S0/0/0
S0/0/1
Site A
R1
Site B
R3
G0/0
G0/1
Filters traffic from 192.168.10.0/24 only to 192.168.30.0/24.
G0/0
G0/1
192.168.10.0/24
192.168.30.0/24
192.168.31.0/24
S1
S2
S3
S4
192.168.11.0/24
PC1
PC2
PC3
PC4

- This example demonstrates the proper placement of the standard ACL that is configured to block traffic from the 192.168.10.0/24 network to the 192.168.30.0/24 network.
- There are two possible places to configure the access-list on R3.
- If the access-list is applied to the S0/0/1 interface, it will block traffic to the 192.168.30.0/24 network, **but also**, going to the 192.168.31.0/24 network.
- The best place to apply the access list is on R3’s G0/0 interface. The access-list list should be applied to traffic exiting the G0/0 interface. Packets from 192.168.10.0/24 can still reach 192.168.31.0/24.

Standard ACL Placement Example

In the figure, the administrator wants to prevent traffic originating in the 192.168.10.0/24 network from reaching the 192.168.30.0/24 network.

If the standard ACL is placed on the outbound interface of R1 (not shown in figure), this would prevent traffic on the 192.168.10.0/24 network from reaching any networks that are reachable through the Serial 0/0/0 interface of R1.

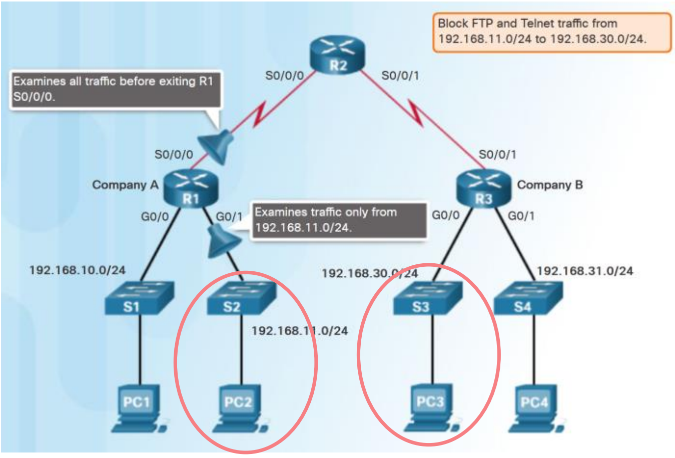
Following the basic placement guidelines of placing the standard ACL close to the destination, the figure shows two possible interface on R3 to apply the standard ACL:

R3 S0/0/1 interface - Applying a standard ACL to prevent traffic from 192.168.10.0/24 from entering the S0/0/1 interface will prevent this traffic from reaching 192.168.30.0/24 and all other networks that are reachable by R3. This includes the 192.168.31.0/24 network. Because the intent of the ACL is to filter traffic destined only for 192.168.30.0/24, a standard ACL should not be applied to this interface.

R3 G0/0 interface - Applying the standard ACL to traffic exiting the G0/0 interface will filter packets from 192.168.10.0/24 to 192.168.30.0/24. This will not affect other networks that are reachable by R3. Packets from 192.168.10.0/24 will still be able to reach 192.168.31.0/24.

Guidelines for ACL Creation

Extended ACL Placement Example


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.

- This example demonstrates the proper placement of the extended ACL that is configured to block Telnet and FTP traffic from the 192.168.11.0/24 network to the 192.168.30.0/24 network.
- There are several ways to accomplish these goals. An extended ACL on R3 that blocks Telnet and FTP from the .11 network would accomplish the task, but the administrator does not control R3. In addition, this solution also allows unwanted traffic to cross the entire network, only to be blocked at the destination. This affects overall network efficiency.
- A better solution is to place an extended ACL on R1 that specifies both source and destination addresses (.11 network and .30 network, respectively), and enforces the rule, "Telnet and FTP traffic from the .11 network is not allowed to go to the .30 network."
- The basic rule for placing an extended ACL is to place it as close to the source as possible.

Extended ACL Placement Example

The basic rule for placing an extended ACL is to place it as close to the source as possible. This prevents unwanted traffic from being sent across multiple networks only to be denied when it reaches its destination. However, network administrators can only place ACL on devices that they control. Therefore, placement must be determined in the context of where the control of the network administrator extends.

In the figure, the administrator of Company A, which includes the 192.168.10.0/24 and 192.168.11.0/24 networks (referred to as .10 and .11 in this example) wants to control traffic to Company B. Specifically, the administrator wants to deny Telnet and FTP traffic from the .11 network to Company B’s 192.168.30.0/24 (.30, in this example) network. At the same time, all other traffic from the .11 network must be permitted to leave Company A without restriction.

There are several ways to accomplish these goals. An extended ACL on R3 that blocks Telnet and FTP from the .11 network would accomplish the task, but the administrator does not control R3. In addition, this solution also allows unwanted traffic to cross the entire network, only to be blocked at the destination. This affects overall network efficiency.

A better solution is to place an extended ACL on R1 that specifies both source and destination addresses (.11 network and .30 network, respectively), and enforces the rule, "Telnet and FTP traffic from the .11 network is not allowed to go to the .30 network." The figure shows two possible interfACE on R1 to apply the extended ACL:

R1 S0/0/0 interface (outbound) - One possibility is to apply an extended ACL outbound on the S0/0/0 interface. Because the extended ACL can examine both source and destination addresses, only FTP and Telnet packets from 192.168.11.0/24 will be denied. Other traffic from 192.168.11.0/24 and other networks will be forwarded by R1. The disadvantage of placing the extended ACL on this interface is that all traffic exiting S0/0/0 must be processed by the ACL including packets from 192.168.10.0/24.

R1 G0/1 interface (inbound) - Applying an extended ACL to traffic entering the G0/1 interface means that only packets from the 192.168.11.0/24 network are subject to ACL processing on R1. Because the filter is to be limited to only those packets leaving the 192.168.11.0/24 network, applying the extended ACL to G0/1 is the best solution.

Guidelines for ACL Creation

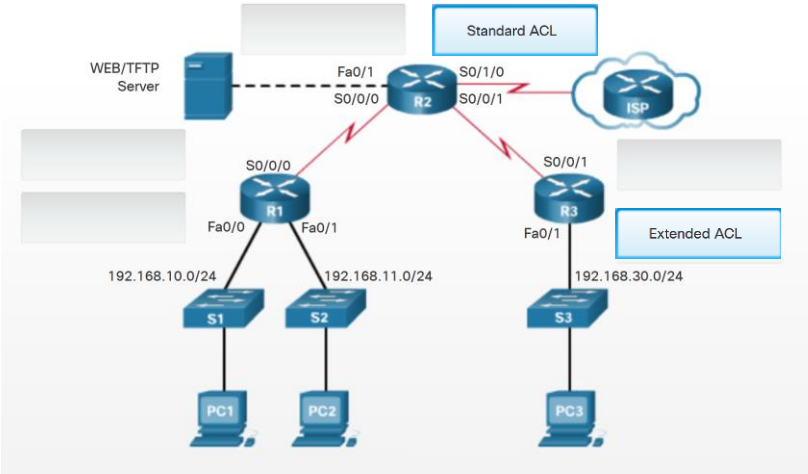
Activity – Placing Standard and Extended ACL

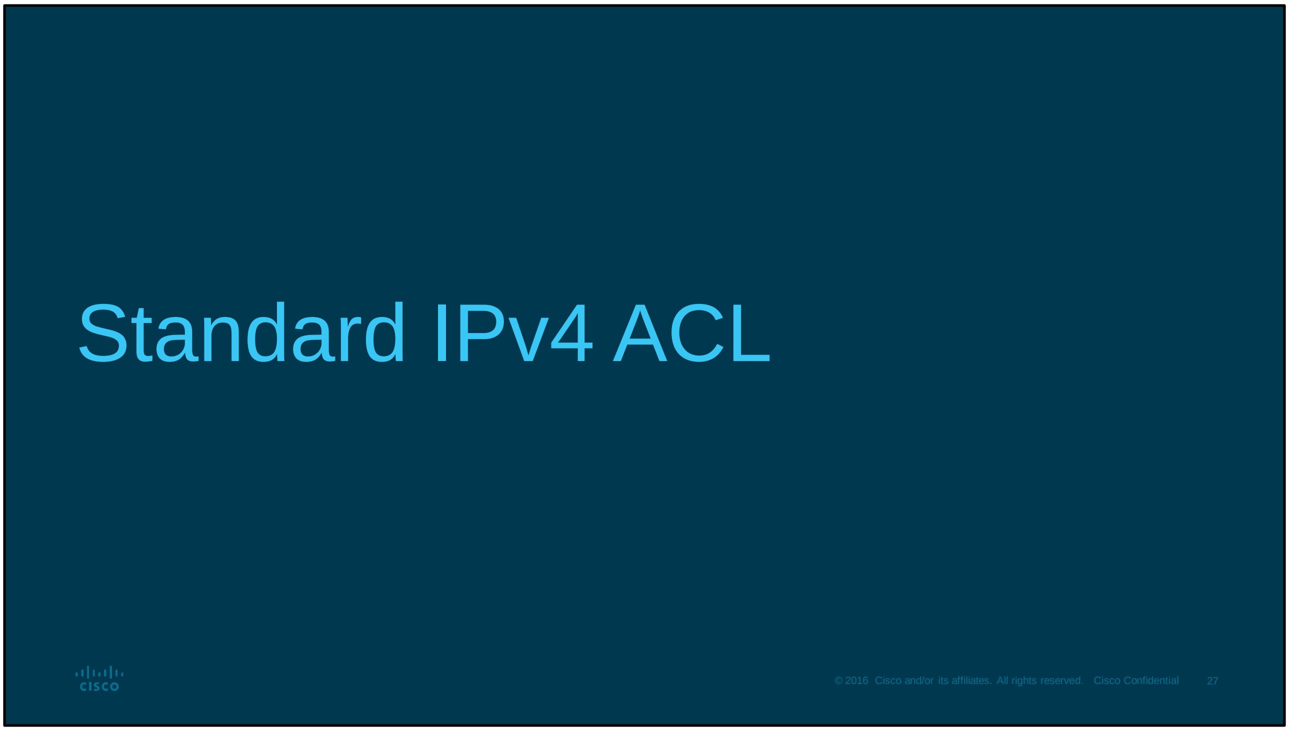
Network Policy #1: Use a standard ACL to stop the 192.168.10.0/24 network from accessing the Internet through ISP.

Network Policy #2: Use an extended ACL to stop the 192.168.30.0/24 network from accessing the Web/TFTP Server.

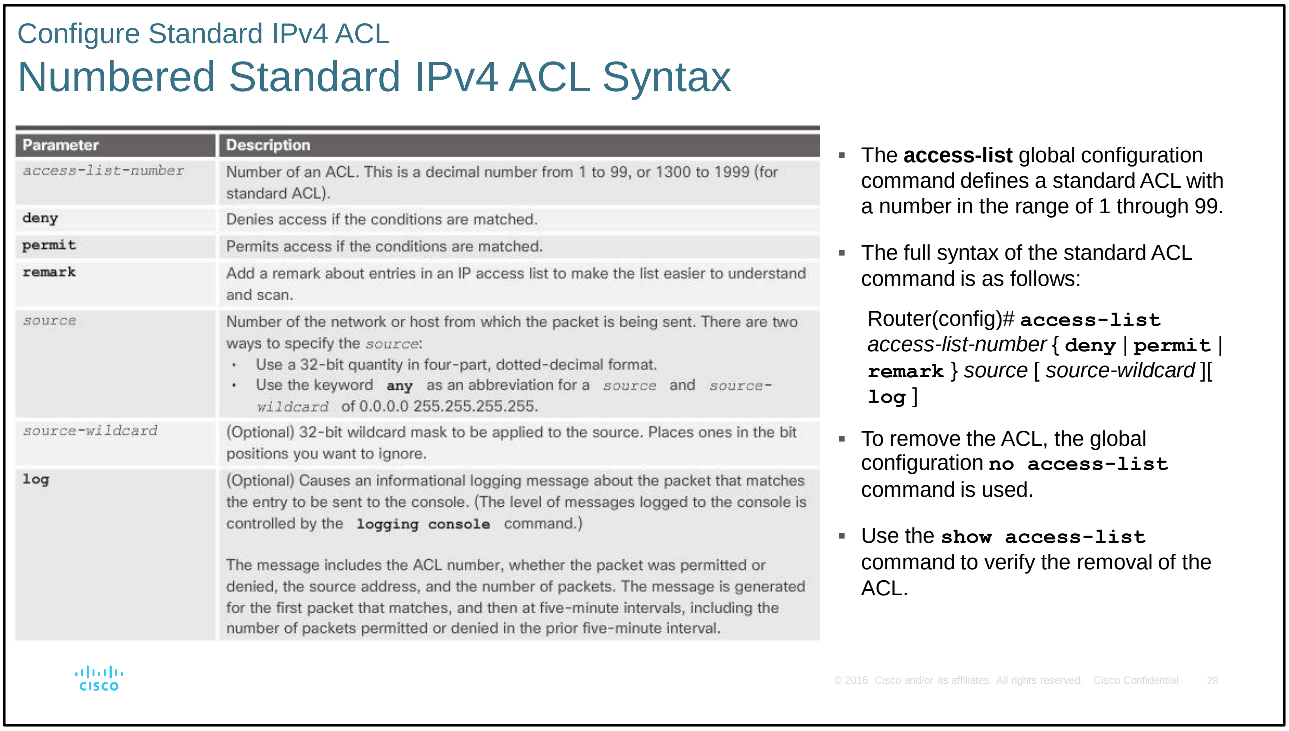
Standard ACL

Extended ACL





Standard IPv4 ACL



Configure Standard IPv4 ACL

Numbered Standard IPv4 ACL Syntax

Parameter	Description
access-list-number	Number of an ACL. This is a decimal number from 1 to 99, or 1300 to 1999 (for standard ACL).
deny	Denies access if the conditions are matched.
permit	Permits access if the conditions are matched.
remark	Add a remark about entries in an IP access list to make the list easier to understand and scan.
source	Number of the network or host from which the packet is being sent. There are two ways to specify the *source*: • Use a 32-bit quantity in four-part, dotted-decimal format. • Use the keyword **any** as an abbreviation for a *source* and *source-wildcard* of 0.0.0.0 255.255.255.255.
source-wildcard	(Optional) 32-bit wildcard mask to be applied to the source. Places ones in the bit positions you want to ignore.
log	(Optional) Causes an informational logging message about the packet that matches the entry to be sent to the console. (The level of messages logged to the console is controlled by the **logging console** command.) The message includes the ACL number, whether the packet was permitted or denied, the source address, and the number of packets. The message is generated for the first packet that matches, and then at five-minute intervals, including the number of packets permitted or denied in the prior five-minute interval.

- 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** | **remark** } *source* [*source-wildcard*][**log**]
- To remove the ACL, the global configuration **no access-list** command is used.
- Use the **show access-list** command to verify the removal of the ACL.

Numbered Standard IPv4 ACL Syntax

To use numbered standard ACL 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. Cisco IOS Software Release 12.0.1 extended these numbers by allowing 1300 to 1999 to be used for standard ACL. This allows for a maximum of 798 possible standard ACL. These additional numbers are referred to as expanded IPv4 ACL.

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**]

Figure 1 provides a detailed explanation of the syntax for a standard ACL.

ACE can permit or deny an individual host or a range of host addresses. To create a host statement in numbered ACL 10 that permits a specific host with the IPv4 address 192.168.10.10, you would enter:

R1(config)# **access-list 10 permit host 192.168.10.10**

As shown in Figure 2, 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:

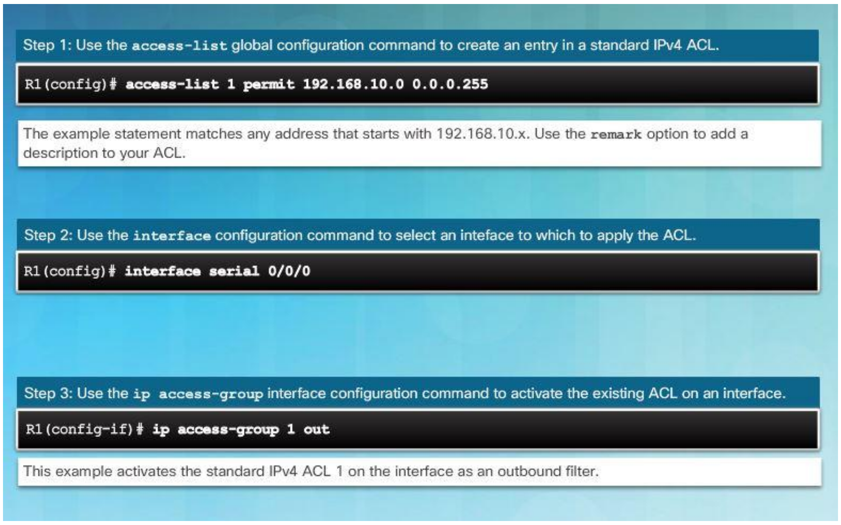
R1(config)# **access-list 10 permit 192.168.10.0 0.0.0.255**

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.

Typically, when an administrator creates an ACL, the purpose of each statement is known and understood. However, to ensure that the administrator and others recall the purpose of a statement, remarks should be included. The **remark** keyword is used for documentation and makes access lists a great deal easier to understand. Each remark is limited to 100 characters. The ACL in Figure 3, although fairly simple, is used to provide an example. When reviewing the ACL in the configuration using the **show running-config** command, the remark is also displayed.

Configure Standard IPv4 ACL

Applying Standard IPv4 ACL to InterfACE



Step 1: Use the `access-list` global configuration command to create an entry in a standard IPv4 ACL.

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

The example statement matches any address that starts with 192.168.10.x. Use the **remark** option to add a description to your ACL.

Step 2: Use the `interface` configuration command to select an inteface to which to apply the ACL.

```
R1(config)# interface serial 0/0/0
```

Step 3: Use the `ip access-group` interface configuration command to activate the existing ACL on an interface.

```
R1(config-if)# ip access-group 1 out
```

This example activates the standard IPv4 ACL 1 on the interface as an outbound filter.

- After a standard IPv4 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 IPv4 ACL to InterfACE

After a standard IPv4 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.

Figure 1 lists the steps and syntax to configure and apply a numbered standard ACL on a router.

Figure 2 shows an example of an ACL designed to permit a single network.

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.

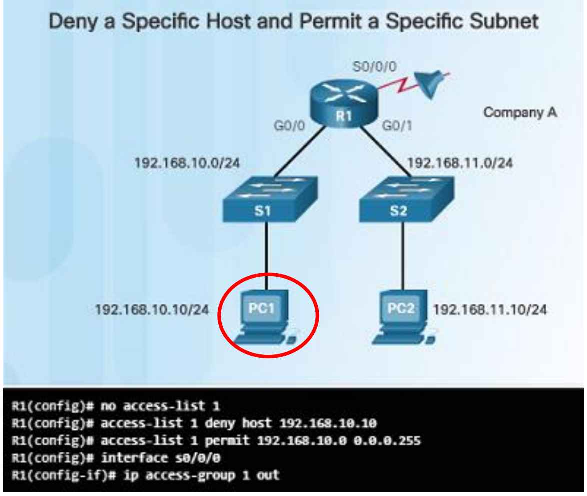
The first line identifies the ACL as access list 1. It permits traffic that matches the selected parameters. In this case, the IPv4 address and wildcard mask identifying the source network is 192.168.10.0 0.0.0.255. Recall that there is an implicit deny all statement that is equivalent to adding the line **access-list 1 deny 0.0.0.0 255.255.255.255** or **access-list deny any** to the end of the ACL.

The **ip access-group 1 out** interface configuration command links and ties ACL 1 to the Serial 0/0/0 interface as an outbound filter.

Therefore, ACL 1 only permits hosts from the 192.168.10.0/24 network to exit router R1. It denies any other network including the 192.168.11.0 network

Configure Standard IPv4 ACL

Numbered Standard IPv4 ACL Examples



```
R1(config)# no access-list 1
R1(config)# access-list 1 deny host 192.168.10.10
R1(config)# access-list 1 permit 192.168.10.0 0.0.0.255
R1(config)# interface s0/0/0
R1(config-if)# ip access-group 1 out
```

- The figure to the left shows an example of an ACL that permits traffic from a specific subnet but denies traffic from a specific host on that subnet.
  - The **`no access-list 1`** command deletes the previous version of ACL 1.
  - The next ACL statement denies the host 192.168.10.10.
  - What is another way to write this command without using **`host`**?
  - All other hosts on the 192.168.10.0/24 network are then permitted.
  - There is an implicit deny statement that matches every other network.
  - Next, the ACL is reapplied to the interface in an outbound direction.

Numbered Standard IPv4 ACL Examples

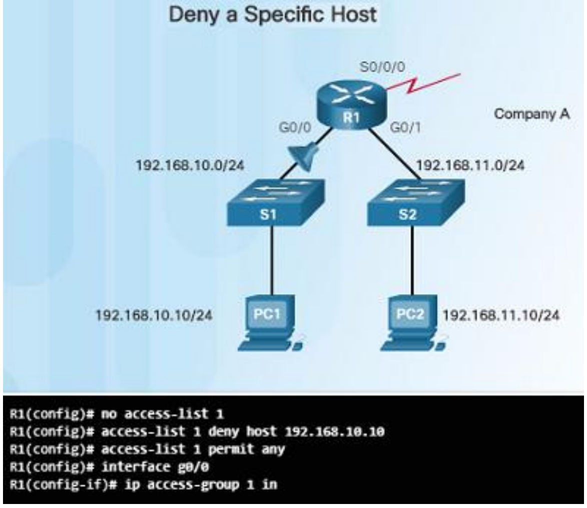
Figure shows an example of an ACL that permits a specific subnet except for a specific host on that subnet.

The first command deletes the previous version of ACL 1. The next ACL statement, denies the PC1 host located at 192.168.10.10. Every other host on the 192.168.10.0/24 network is then permitted. Again the implicit deny statement matches every other network.

The ACL is reapplied to interface S0/0/0 in an outbound direction.

Configure Standard IPv4 ACL

Numbered Standard IPv4 ACL Examples (Cont.)



```
R1(config)# no access-list 1
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
```

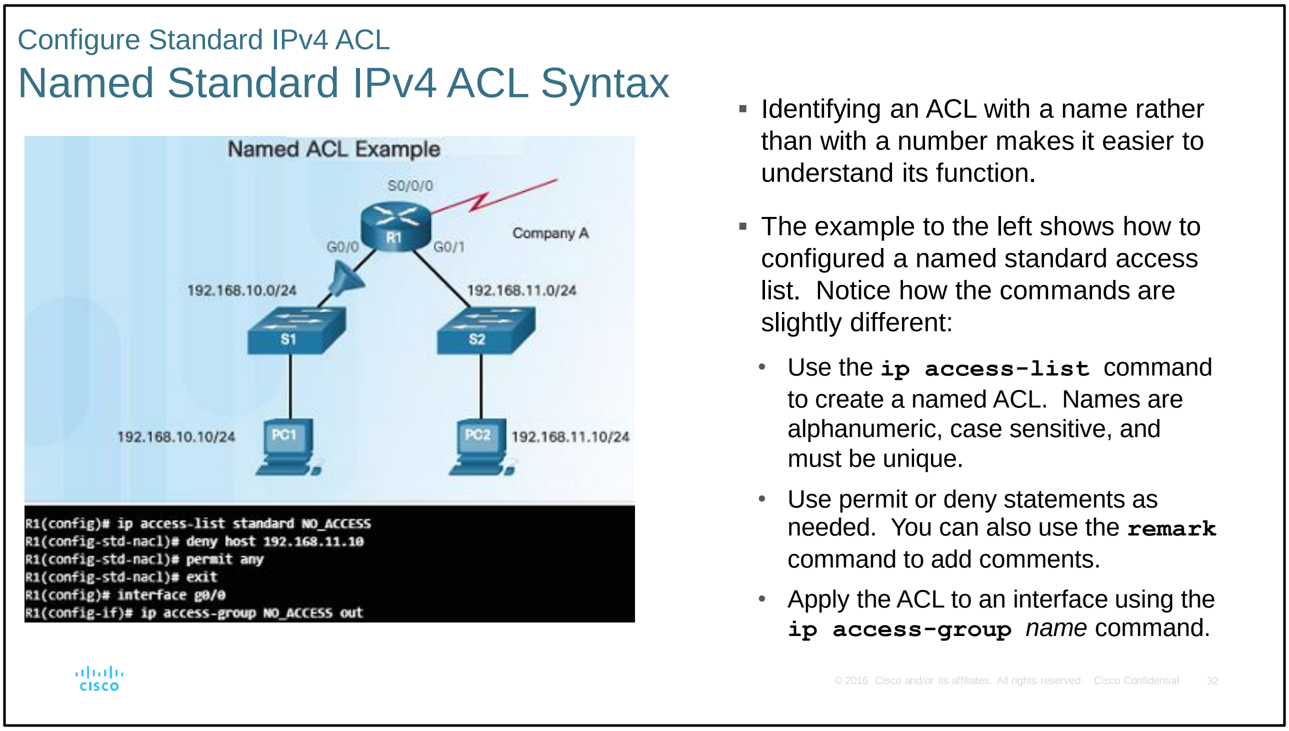
- This next example demonstrates an ACL that denies a specific host but will permit all other traffic.
  - The first ACL statement deletes the previous version of ACL 1.
  - The next command, with the deny keyword, will deny traffic from the PC1 host that is located at 192.168.10.10.
  - The **`access-list 1 permit any`** statement will permit all other hosts.
  - This ACL is applied to interface G0/0 in the inbound direction since it only affects the 192.168.10.0/24 LAN.

Figure shows an example of an ACL that denies a specific host. This ACL replACE the previous example. This example still blocks traffic from host PC1 but permits all other traffic.

The first two commands are the same as the previous example. The first command deletes the previous version of ACL 1 and the next ACL statement denies the PC1 host that is located at 192.168.10.10.

The third line is new and permits all other hosts. This means that all hosts from the 192.168.10.0/24 network will be permitted except for PC1, which was denied in the previous statement.

This ACL is applied to interface G0/0 in the inbound direction. Because the filter only affects the 192.168.10.0/24 LAN on G0/0 it is more efficient to apply the ACL to the inbound interface. The ACL could be applied to S0/0/0 in the outbound direction but then R1 would have to examine packets from all networks including 192.168.11.0/24.



Configure Standard IPv4 ACL

Named Standard IPv4 ACL Syntax

```
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
```

- Identifying an ACL with a name rather than with a number makes it easier to understand its function.
- The example to the left shows how to configured a named standard access list. Notice how the commands are slightly different:
  - Use the `ip access-list` command to create a named ACL. Names are alphanumeric, case sensitive, and must be unique.
  - Use permit or deny statements as needed. You can also use the `remark` command to add comments.
  - Apply the ACL to an interface using the `ip access-group` *name* command.

Named Standard IPv4 ACL Syntax

Naming an ACL makes it easier to understand its function. When you identify your ACL with a name instead of with a number, the configuration mode and command syntax are slightly different.

Figure 1 shows the steps required to create a standard named ACL.

Step 1. Starting from the global configuration mode, use the **ip access-list** command to create a named ACL. ACL names are alphanumeric, case sensitive, and must be unique. The **ip access-list standard** *name* command is used to create a standard named ACL. After entering the command, the router is in standard (std) named ACL (nacl) configuration mode as indicated by the second prompt in the Figure 1.

Note: Numbered ACL use the global configuration command **access-list**, whereas named IPv4 ACL use the **ip access-list** command.

Step 2. From the named ACL configuration mode, use **permit** or **deny** statements to specify one or more conditions for determining whether a packet is forwarded or dropped. You can use **remark** to add a comment to the ACL.

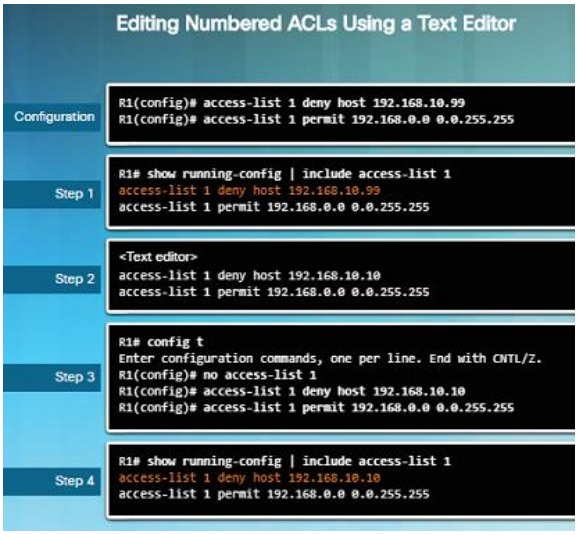
Step 3. Apply the ACL to an interface using the **ip access-group** *name* command. Specify whether the ACL should be applied to packets as they enter the interface (**in**) or applied to packets as they exit the interface (**out**).

Figure 2 shows the commands used to configure a standard named ACL on router R1, interface G0/0, that denies host 192.168.11.10 access to the 192.168.10.0 network. The ACL is named NO_ACCESS.

Capitalizing ACL names is not required, but makes them stand out when viewing the running-config output. It also makes it less likely that you will accidentally create two different ACL with the same name but with different uses of capitalization.

Verify and Modify IPv4 ACL

Modify an ACL: Method 1 – Use a Text Editor



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
```

- It is sometimes easier to create and edit ACL in a text editor such as Microsoft Notepad rather making changes directly on the router.
- For an existing ACL, use the **`show running-config`** command to display the ACL, copy and paste it into the text editor, make the necessary changes, and then paste it back into the router interface.
- It is important to note that when using the no access-list command, different IOS software releases act differently.
  - If the ACL that has been deleted is still applied to the interface, some IOS versions act as if no ACL is protecting your network while others deny all traffic.

Method 1 - Use a Text Editor

After someone is familiar with creating and editing ACL, it may be easier to construct the ACL using a text editor such as Microsoft Notepad. This allows you to create or edit the ACL and then paste it into the router interface. For an existing ACL, you can use the **show running-config** command to display the ACL, copy and paste it into the text editor, make the necessary changes, and paste it back in to the router interface.

Configuration: For example, assume that the host IPv4 address in the figure was incorrectly entered. Instead of the 192.168.10.99 host, it should have been the 192.168.10.10 host. Here are the steps to edit and correct ACL 1:

Step 1. Display the ACL using the **show running-config** command. The example in the figure uses the **include** keyword to display only the ACE.

Step 2. Highlight the ACL, copy it, and then paste it into Microsoft Notepad. Edit the list as required. After the ACL is correctly displayed in Microsoft Notepad, highlight it and copy it.

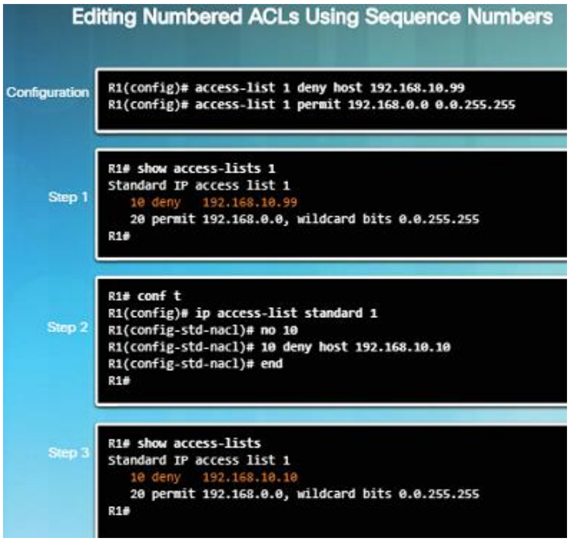
Step 3. In global configuration mode, remove the access list using the **no access-list 1** command. Otherwise, the new statements would be appended to the existing ACL. Then paste the new ACL into the configuration of the router.

Step 4. Using the **show running-config** command, verify the changes

It should be mentioned that when using the **no access-list** command, different IOS software releases act differently. If the ACL that has been deleted is still applied to an interface, some IOS versions act as if no ACL is protecting your network while others deny all traffic. For this reason it is good practice to remove the reference to the access list from the interface before modifying the access list. If there is an error in the new list, disable it and troubleshoot the problem. In that instance, the network has no ACL during the correction process.

Verify and Modify IPv4 ACL

Modify an ACL: Method 2 – Use Sequence Numbers in named ACL



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#
```

- The figure to the left demonstrates the steps used to make changes to a numbered ACL using sequence numbers.
- Step 1 identifies the problem. The **`deny 192.168.10.99`** statement is incorrect. The host to deny should be 192.168.10.10
- To make the edit, Step 2 shows how to go into standard access-list 1 and make the change. The misconfigured statement had to be deleted with the no command: **`no 10`**
- Once it was deleted, the new statement with the correct host was added: **`10 deny host 192.168.10.10`**

Method 2 - Use Sequence Numbers

As shown in the figure, the initial configuration of ACL 1 included a host statement for host 192.168.10.99. This was in error. The host should have been configured as 192.168.10.10. To edit the ACL using sequence numbers follow these steps:

Step 1. Display the current ACL using the **show access-lists 1** command. The output from this command will be discussed in more detail later in this section. The sequence number is displayed at the beginning of each statement. The sequence number was automatically assigned when the access list statement was entered. Notice that the misconfigured statement has the sequence number 10.

Step 2. Enter the **ip access-lists standard** command that is used to configure named ACL. The ACL number 1, is used as the name. First, the misconfigured statement needs to be deleted using the **no 10** command with 10 referring to the sequence number. Next, a new sequence number 10 statement is added using the command, **10 deny host 192.168.10.10**.

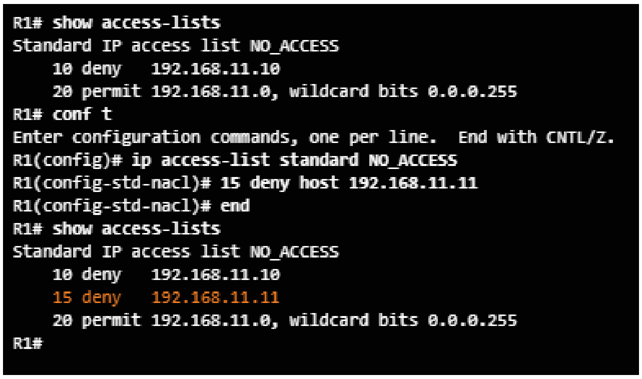
Note: Statements cannot be overwritten using the same sequence number as an existing statement. The current statement must be deleted first, and then the new one can be added.

Step 3. Verify the changes using the **show access-lists** command.

As discussed previously, Cisco IOS implements an internal logic to standard access lists. The order in which standard ACE are entered may not be the order in which they are stored, displayed or processed by the router. The **show access-lists** command displays the ACE with their sequence numbers.

Verify and Modify IPv4 ACL

Editing Standard 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#
```

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

- By referring to statement sequence numbers, individual statements can be easily inserted or deleted.
- The figure to the left shows an example of how to insert a line into a named ACL.
- By numbering it 15, it will place the command in between statement 10 and 20.
- Please notice that when the ACL was originally created, the network administrator spaced each command by 10 which left room for edits and additions.

Editing Standard Named ACL

In a previous example, sequence numbers were used to edit a standard numbered IPv4 ACL. By referring to the statement sequence numbers, individual statements can easily be inserted or deleted. This method can also be used to edit standard named ACL.

The figure shows an example of inserting a line to a named ACL.

In the first **show** command output, you can see that the ACL named NO_ACCESS has two numbered lines indicating access rules for a workstation with the IPv4 address 192.168.11.10.

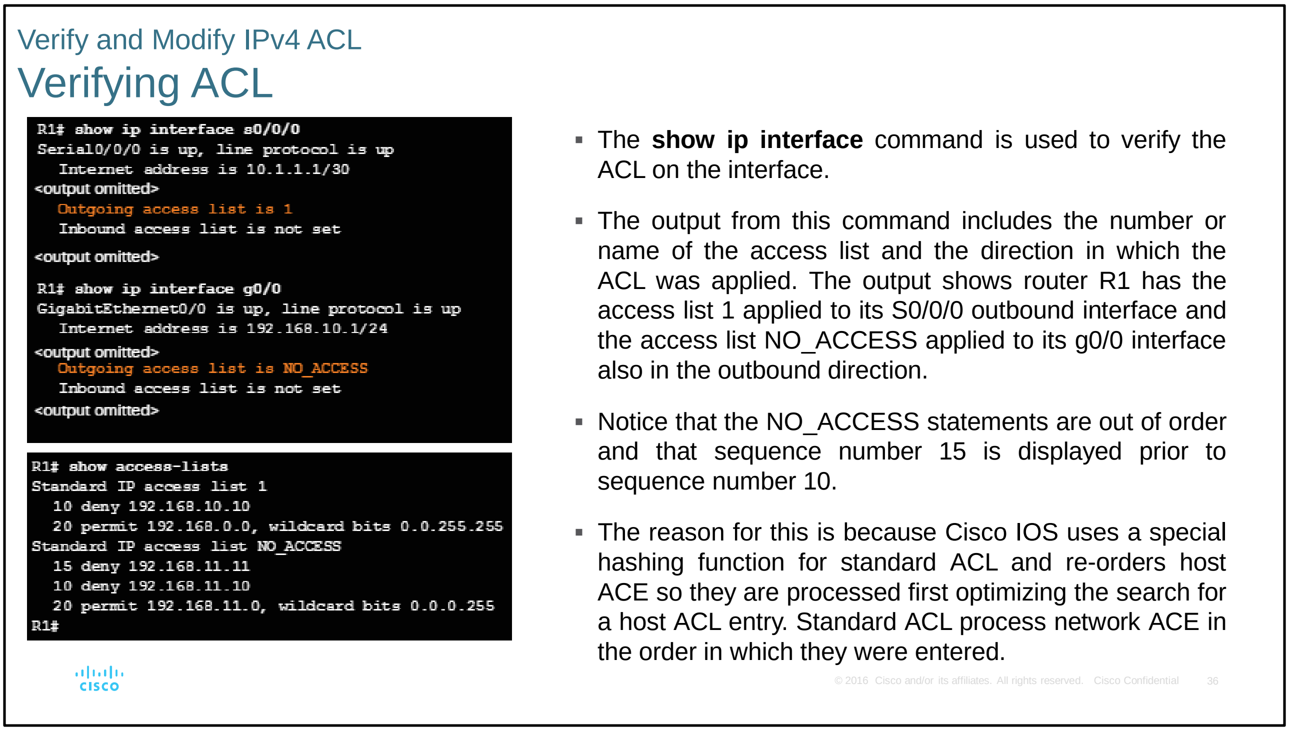
From named access list configuration mode, statements can be inserted or removed.

To add a statement to deny another workstation requires inserting a numbered line.

In the example, the workstation with the IPv4 address 192.168.11.11 is being added using a new sequence number of 15.

The final **show** command output verifies that the new workstation is now denied access.

Note: In named access list configuration mode, use the **no** *sequence-number* command to quickly delete individual statements.



Verify and Modify IPv4 ACL

Verifying ACL

```
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#
```

- The **show ip interface** command is used to verify the ACL on the interface.
- The output from this command includes the number or name of the access list and the direction in which the ACL was applied. The output shows router R1 has the access list 1 applied to its S0/0/0 outbound interface and the access list NO_ACCESS applied to its g0/0 interface also in the outbound direction.
- Notice that the NO_ACCESS statements are out of order and that sequence number 15 is displayed prior to sequence number 10.
- The reason for this is because Cisco IOS uses a special hashing function for standard ACL and re-orders host ACE so they are processed first optimizing the search for a host ACL entry. Standard ACL process network ACE in the order in which they were entered.

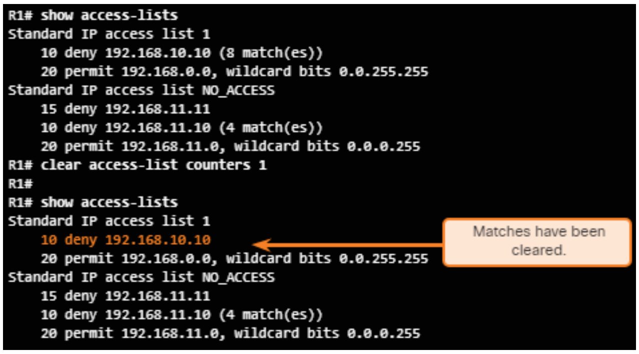
Verifying ACL

As shown in Figure, the **show ip interface** command is used to verify the ACL on the interface. The output from this command includes the number or name of the access list and the direction in which the ACL was applied. The output shows router R1 has the access list 1 applied to its S0/0/0 outbound interface and the access list NO_ACCESS applied to its g0/0 interface, also in the outbound direction.

The example in Figure shows also the result of issuing the **show access-lists** command on router R1. To view an individual access list use the **show access-lists** command followed by the access list number or name. The NO_ACCESS statements may look strange. Notice that sequence number 15 is displayed prior to sequence number 10. This is a result of the router’s internal process and will be discussed later in this section.

Verify and Modify IPv4 ACL

ACL Statistics



```
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# clear access-list counters 1
R1#
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 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
```

Matches have been cleared.

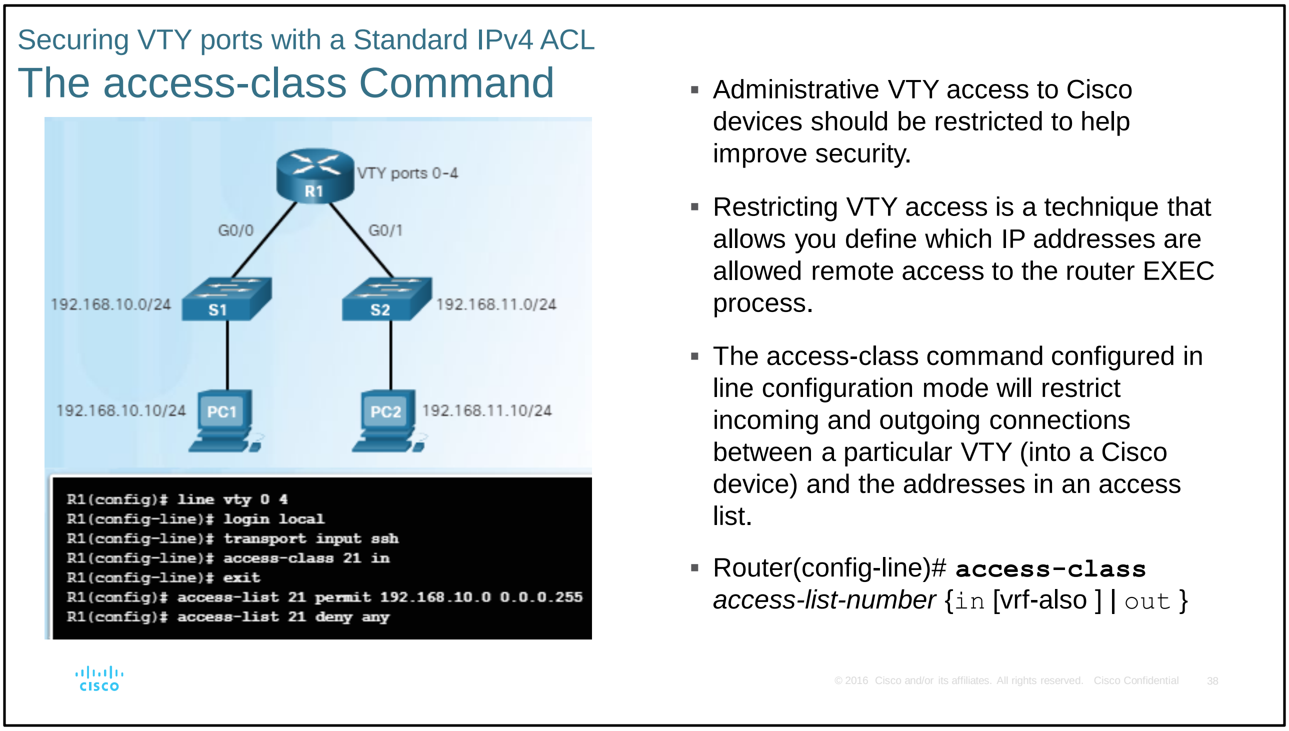
- The **show access-lists** command can be used to display matched statistics after an ACL has been applied to an interface and some testing has occurred.
- When traffic is generated that should match an ACL statement, the matches shown in the **show access-lists** command output should increase.
- Recall that every ACL has an implicit **deny any** as the last statement. The statistics for this implicit command will not be displayed. However, if this command is configured manually, the results will be displayed.
- The clear access-list counters command can be used to clear the counters for testing purposes.

ACL Statistics

After an ACL has been applied to an interface and some testing has occurred, the **show access-lists** command will show statistics for each statement that has been matched. In the output in Figure 1, note that some of the statements have been matched. When traffic is generated that should match an ACL statement, the matches shown in the **show access-lists** command output should increase. For instance, in this example, if a ping is issued from PC1 to PC3 or PC4, the output will show an increase in the matches for the deny statement of ACL 1.

Both permit and deny statements will track statistics for matches; however, recall that every ACL has an implied deny any as the last statement. This statement will not appear in the **show access-lists** command; therefore, statistics for that statement will not appear. To view statistics for the implied deny any statement, the statement can be configured manually and will appear in the output.

During testing of an ACL, the counters can be cleared using the **clear access-list counters** command. This command can be used alone or with the number or name of a specific ACL. As shown in Figure 2, this command clears the statistic counters for an ACL.



Securing VTY ports with a Standard IPv4 ACL

The access-class Command

```
R1(config)# line vty 0 4
R1(config-line)# login local
R1(config-line)# transport input ssh
R1(config-line)# access-class 21 in
R1(config-line)# exit
R1(config)# access-list 21 permit 192.168.10.0 0.0.0.255
R1(config)# access-list 21 deny any
```

- Administrative VTY access to Cisco devices should be restricted to help improve security.
- Restricting VTY access is a technique that allows you define which IP addresses are allowed remote access to the router EXEC process.
- The access-class command configured in line configuration mode will restrict incoming and outgoing connections between a particular VTY (into a Cisco device) and the addresses in an access list.
- Router(config-line)# **`access-class`** *access-list-number* {`in` [vrf-also] | `out` }

The access-class Command

You can improve the security of administrative lines by restricting VTY access. Restricting VTY access is a technique that allows you to define which IP addresses are allowed remote access to the router EXEC process. You can specify which IP addresses are allowed remote access to your router with an ACL and an **access-class** statement configured on your VTY lines. Use this technique with SSH to further improve administrative access security.

The **access-class** command configured in line configuration mode restricts incoming and outgoing connections between a particular VTY (into a Cisco device) and the addresses in an access list.

The command syntax of the **access-class** command is:

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

The parameter **in** restricts incoming connections between the addresses in the access list and the Cisco device, while the parameter **out** restricts outgoing connections between a particular Cisco device and the addresses in the access list.

An example allowing a range of addresses to access VTY lines 0 - 4 is shown in Figure. The ACL in the figure is configured to permit network 192.168.10.0 to access VTY lines 0 - 4 but deny all other networks.

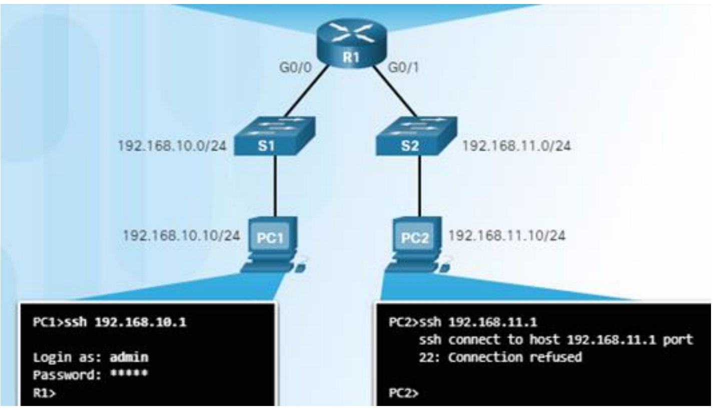
The following should be considered when configuring access lists on VTY:

- Both named and numbered access lists can be applied to VTY.
- Identical restrictions should be set on all the VTY, because a user can attempt to connect to any of them.

Note: Access lists apply to packets that travel through a router. They are not designed to block packets that originate within the router. By default, an outbound ACL does not prevent remote access connections initiated from the router.

Securing VTY ports with a Standard IPv4 ACL

Verifying the VTY Port is Secured

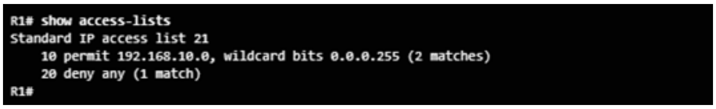


```
PC1>ssh 192.168.10.1

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

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

PC2>
```



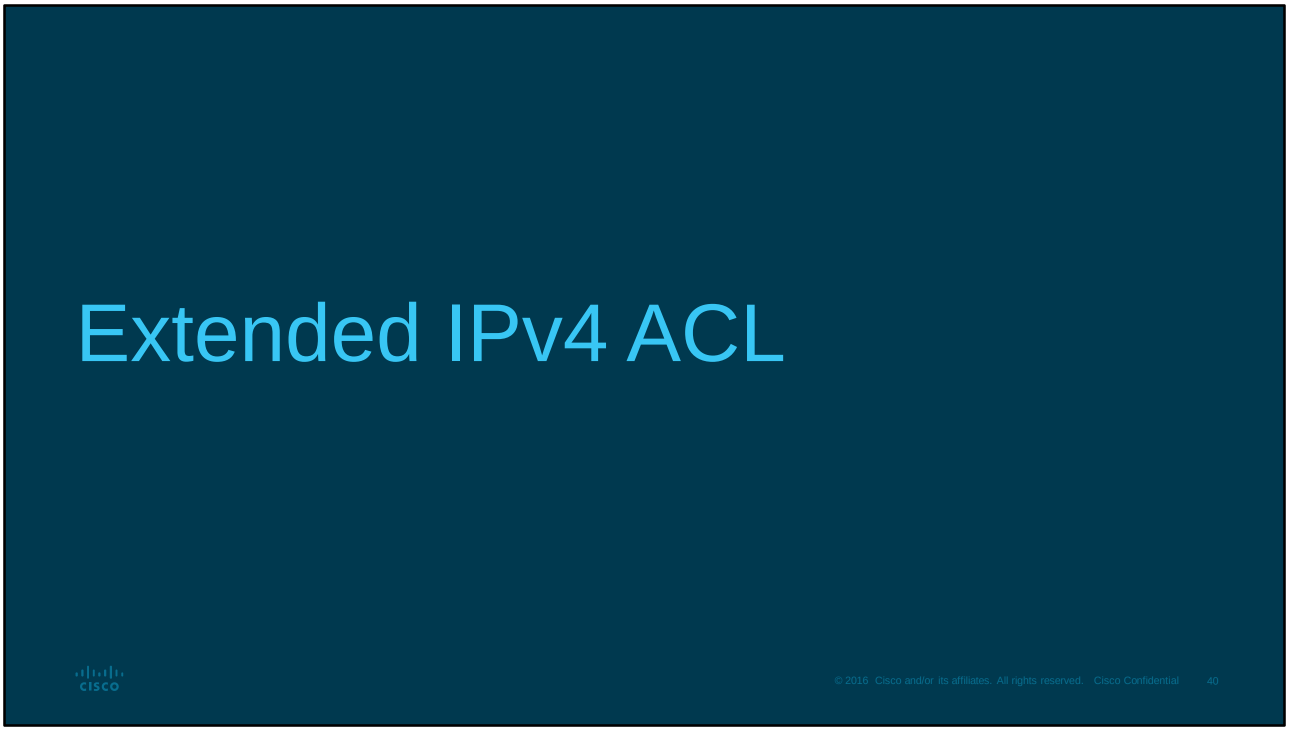
```
R1# show access-lists
Standard IP access list 21
    10 permit 192.168.10.0, wildcard bits 0.0.0.255 (2 matches)
    20 deny any (1 match)
R1#
```

- Verification of the ACL configuration used to restrict VTY access is important.
- The figure to the left shows two devices trying to ssh into two different devices.
- The show access-lists command output shows the results after the SSH attempts by PC1 and PC2.
- Notice the match results in the permit and the deny statements.

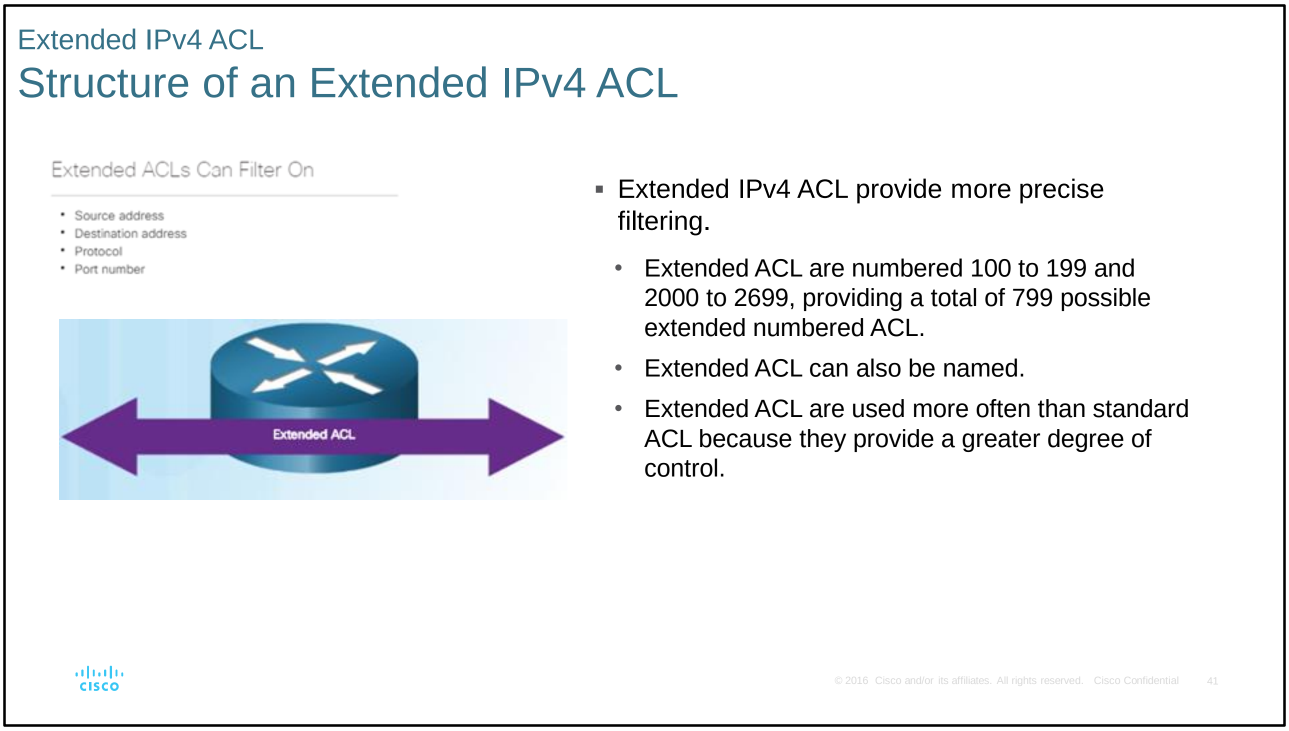
Verifying the VTY Port is Secured

After the ACL to restrict access to the VTY lines is configured, it is important to verify that it is working as expected. The figure shows two devices attempting to connect to R1 using SSH. Access list 21 has been configured on the VTY lines on R1. PC1 is successful while PC2 fails to establish a SSH connection. This is the expected behavior, as the configured access list permits VTY access from the 192.168.10.0/24 network while denying all other devices.

The output for R1 shows the result of issuing the **show access-lists** command after the SSH attempts by PC1 and PC2. The match in the permit line of the output is a result of a successful SSH connection by PC1. The match in the deny statement is due to the failed attempt to create an SSH connection by PC2, a device on the 192.168.11.0/24 network.



Extended IPv4 ACL



Extended IPv4 ACL

Structure of an Extended IPv4 ACL

Extended ACLs Can Filter On

- Source address
- Destination address
- Protocol
- Port number

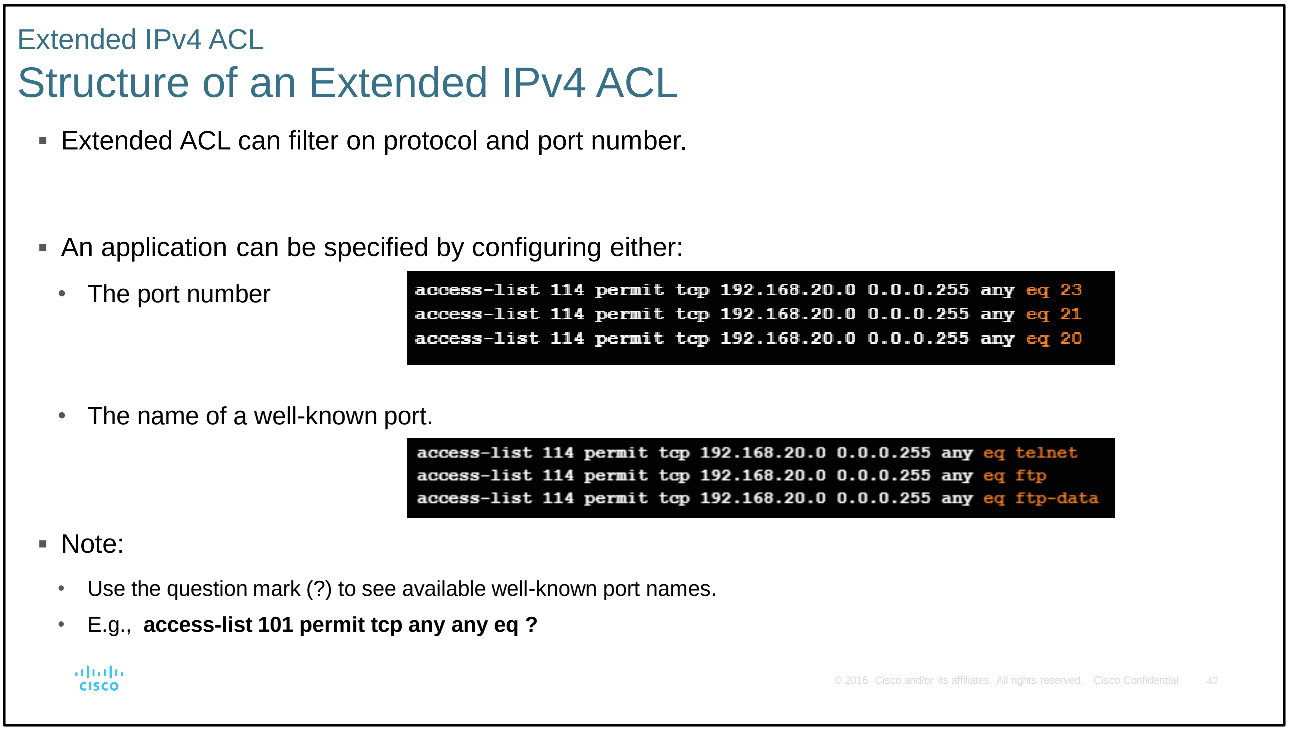
- Extended IPv4 ACL provide more precise filtering.
  - Extended ACL are numbered 100 to 199 and 2000 to 2699, providing a total of 799 possible extended numbered ACL.
  - Extended ACL can also be named.
  - Extended ACL are used more often than standard ACL because they provide a greater degree of control.

Extended ACL

Testing Packets with Extended ACL

For more precise traffic-filtering control, extended IPv4 ACL can be created. Extended ACL are numbered 100 to 199 and 2000 to 2699, providing a total of 799 possible extended numbered ACL. Extended ACL can also be named.

Extended ACL are used more often than standard ACL because they provide a greater degree of control. As shown in the figure, like standard ACL, extended ACL have the ability to check source addresses of packets, but they also have the ability to check the destination address, protocols, and port numbers (or services). This provides a greater range of criteria on which to base the ACL. For example, one extended ACL can allow email traffic from a network to a specific destination while denying file transfers and web browsing.



Extended IPv4 ACL

Structure of an Extended IPv4 ACL

- Extended ACL 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 ?**

Filtering Ports and Services

The ability to filter on protocol and port number allows network administrators to build very specific extended ACL. An application can be specified by configuring either the port number or the name of a well-known port.

Figure 1 shows some examples of how an administrator specifies a TCP or UDP port number by placing it at the end of the extended ACL statement. Logical operations can be used, such as equal (eq), not equal (neq), greater than (gt), and less than (lt).

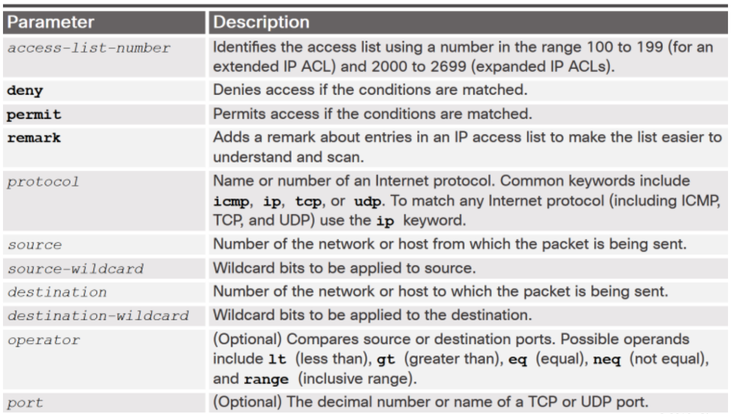
Figure 2 shows how to display a list of port numbers and keywords that can be used when building an ACL using the command:

R1(config)# **access-list 101 permit tcp any any eq ?**

Extended IPv4 ACL

Configure Extended IPv4 ACL

- 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*]]



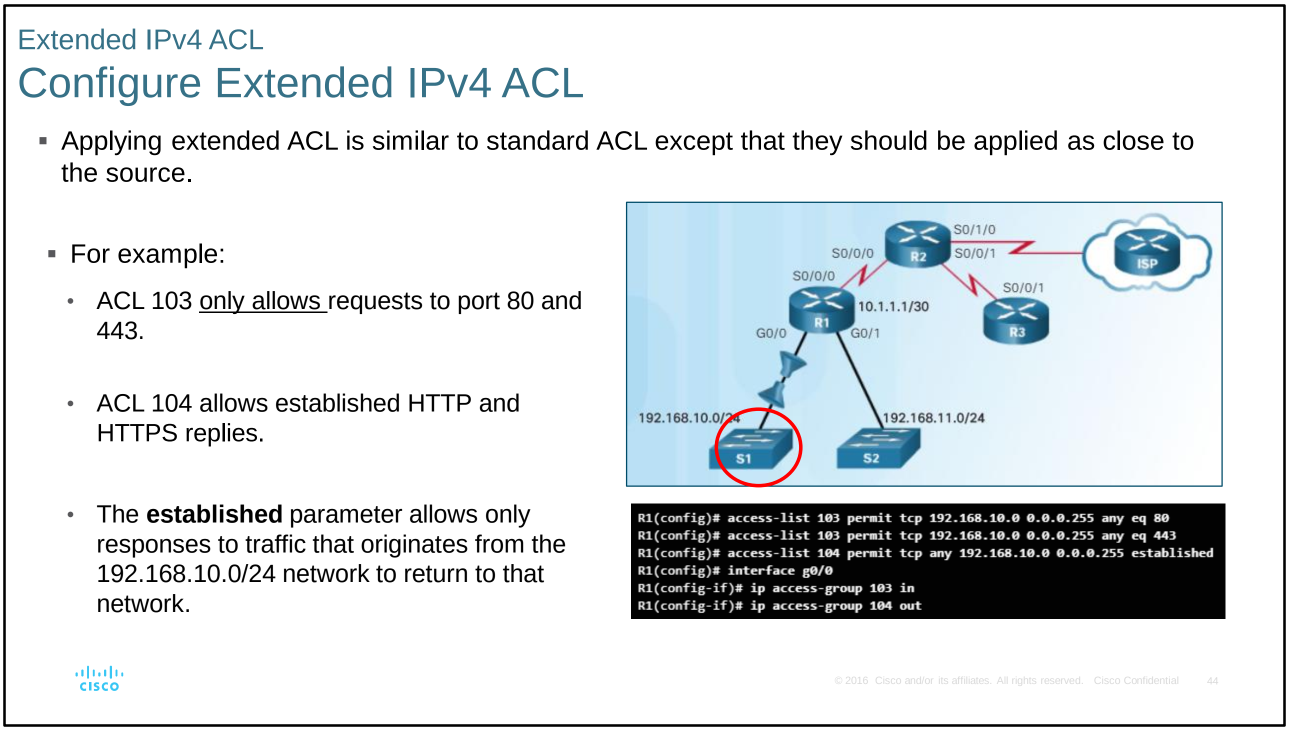
Parameter	Description
access-list-number	Identifies the access list using a number in the range 100 to 199 (for an extended IP ACL) and 2000 to 2699 (expanded IP ACLs).
deny	Denies access if the conditions are matched.
permit	Permits access if the conditions are matched.
remark	Adds a remark about entries in an IP access list to make the list easier to understand and scan.
protocol	Name or number of an Internet protocol. Common keywords include **icmp**, **ip**, **tcp**, or **udp**. To match any Internet protocol (including ICMP, TCP, and UDP) use the **ip** keyword.
source	Number of the network or host from which the packet is being sent.
source-wildcard	Wildcard bits to be applied to source.
destination	Number of the network or host to which the packet is being sent.
destination-wildcard	Wildcard bits to be applied to the destination.
operator	(Optional) Compares source or destination ports. Possible operands include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range).
port	(Optional) The decimal number or name of a TCP or UDP port.

Configuring Extended ACL

The procedural steps for configuring extended ACL are the same as for standard ACL. 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 ACL.

Note: The internal logic applied to the ordering of standard ACL statements does not apply to extended ACL. The order in which the statements are entered during configuration is the order they are displayed and processed.

Figure shows the common command syntax for extended IPv4 ACL. Note that there are many keywords and parameters for extended ACL. It is not necessary to use all of the keywords and parameters when configuring an extended ACL. Recall that the **?** can be used to get help when entering complex commands.



Extended IPv4 ACL

Configure Extended IPv4 ACL

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

- For example:
  - ACL 103 only 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
R1(config)# interface g0/0
R1(config-if)# ip access-group 103 in
R1(config-if)# ip access-group 104 out
```

Figure shows an example of an extended ACL. In this example, the network administrator has configured ACL 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.

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.

The **established** parameter allows only responses to traffic that originates from the 192.168.10.0/24 network to return to that network. A match occurs if the returning TCP segment has the ACK or reset (RST) bits set, which indicates that the packet belongs to an existing connection. 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.

Applying Extended ACL to Interfaces

In the previous example, the network administrator configured an ACL to allow users from the 192.168.10.0/24 network to browse both insecure and secure websites.

Even though it has been configured, the ACL will not filter traffic until it is applied to an interface. To apply an ACL to an interface, first consider whether the traffic to be filtered is going in or out. When a user on the internal LAN accesses a website on the Internet, traffic is traffic going out to the Internet. When an internal user receives an email from the Internet, traffic is coming into the local router. However, when applying an ACL to an interface, in and out take on different meanings. From an ACL consideration, in and out are in reference to the router interface.

In the topology in the figure, R1 has three interfaces. It has a serial interface, S0/0/0, and two Gigabit Ethernet interfaces, G0/0 and G0/1. Recall that an extended ACL should typically be applied close to the source. In this topology the interface closest to the source of the target traffic is the G0/0 interface.

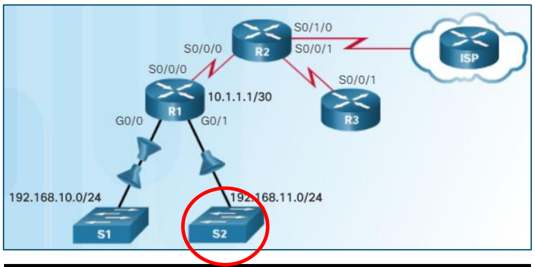
Web request traffic from users on the 192.168.10.0/24 LAN is inbound to the G0/0 interface. Return traffic from established connections to users on the LAN is outbound from the G0/0 interface. The example applies the ACL to the G0/0 interface in both directions. The inbound ACL, 103, checks for the type of traffic. The outbound ACL, 104, checks for return traffic from established connections. This will restrict 192.168.10.0 Internet access to allow only website browsing.

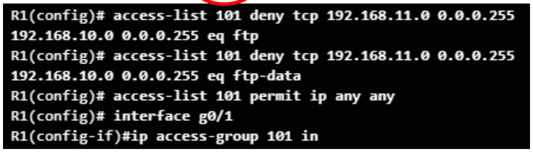
Note: The access lists could have been applied to the S0/0/0 interface but in that case, the router’s ACL process would have to examine all packets entering the router, not only traffic to and from 192.168.11.0. This would cause unnecessary processing by the router.

Extended IPv4 ACL

Configure Extended IPv4 ACL

- 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 ACE 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
```

Filtering Traffic with Extended ACL

The example shown in Figure denies FTP traffic from subnet 192.168.11.0 that is going to subnet 192.168.10.0, but permits all other traffic. 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 101 deny tcp 192.168.11.0 0.0.0.255 192.168.10.0 0.0.0.255 eq 20

access-list 101 deny tcp 192.168.11.0 0.0.0.255 192.168.10.0 0.0.0.255 eq 21

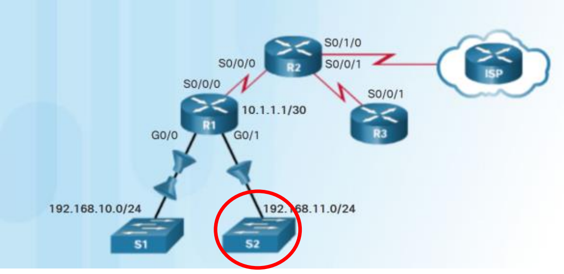
To prevent the implied **deny any** statement at the end of the ACL from blocking all traffic, the **permit ip any any** statement is added. Without at least one **permit** statement in an ACL, all traffic on the interface where that ACL was applied would be dropped. The ACL should be applied inbound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

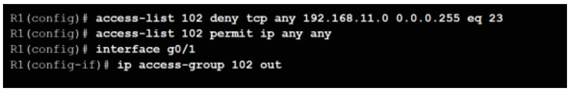
Note: The example use the **permit ip any any** statement at the end of the ACL. For greater security the **permit 192.168.11.0 0.0.0.255 any** command may be used.

Extended IPv4 ACL

Configure Extended IPv4 ACL

- The example denies Telnet traffic from any source to the 192.168.11.0/24 LAN but allows all other IP traffic.
  - Because traffic destined for the 192.168.11.0/24 LAN is outbound on interface G0/1, the ACL would be applied to G0/1 using the **out** keyword.
  - Note the use of the **any** keywords in the permit statement. This permit statement is added to ensure that no other traffic is blocked.





```
R1(config)# access-list 102 deny tcp any 192.168.11.0 0.0.0.255 eq 23
R1(config)# access-list 102 permit ip any any
R1(config)# interface g0/1
R1(config-if)# ip access-group 102 out
```

Filtering Traffic with Extended ACL

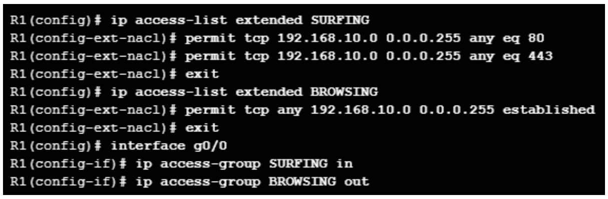
The example shown in Figure, denies Telnet traffic from any source to the 192.168.11.0/24 LAN, but allows all other IP traffic. Because traffic destined for the 192.168.11.0/24 LAN is outbound on interface G0/1, the ACL would be applied to G0/1 using the **out** keyword. Note the use of the **any** keywords in the permit statement. This permit statement is added to ensure that no other traffic is blocked.

Note: The example use the **permit ip any any** statement at the end of the ACL. For greater security the **permit 192.168.11.0 0.0.0.255 any** command may be used.

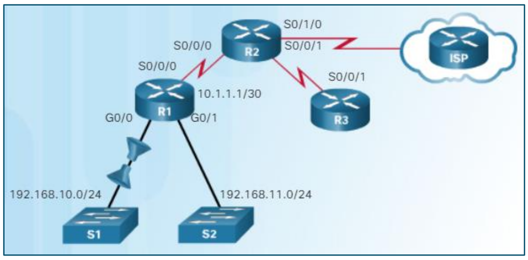
Extended IPv4 ACL

Configure Extended IPv4 ACL

- Named extended ACL are created in the same way that named standard ACL are created.
- In this example, two named ACL 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
```



Creating Named Extended ACL

Named extended ACL are created in essentially the same way that named standard ACL are created. Follow these steps to create an extended ACL, using names:

Step 1. From global configuration mode, use the **ip access-list extended** *name* command to define a name for the extended ACL.

Step 2. In named ACL configuration mode, specify the conditions to **permit** or **deny**.

Step 3. From interface configuration mode, apply the named ACL using the **ip access-group***name* [**in** | **out**] command.

Step 4. Return to privileged EXEC mode and verify the ACL with the **show access-lists** *name* command.

Step 5. Save the entries in the configuration file with the **copy running-config startup-config** command.

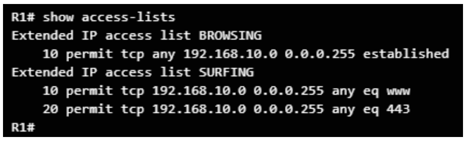
To remove a named extended ACL, use the **no ip access-list extended** *name* global configuration command.

The figure shows the named versions of the ACL created in the previous examples. The named ACL, SURFING, permits the users on the 192.168.10.0/24 LAN to access web sites. The named ACL, BROWSING, allows the return traffic from established connections. Using the ACL names, the rules are applied inbound and outbound on the G0/0 interface.

Extended IPv4 ACL

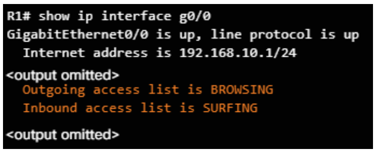
Configure Extended IPv4 ACL

- The **show ip interface** and **show access-lists** commands can be used to verify the content of extended ACL.
- The output and sequence numbers displayed in the **show access-lists** command output is the order in which the statements were entered.
  - Unlike standard ACL, extended ACL 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>
```

Verifying Extended ACL

After an ACL has been configured and applied to an interface, use Cisco IOS **show** commands to verify the configuration. In the figure, the top example shows the Cisco IOS command used to display the contents of all ACL. The bottom example shows the result of issuing the **show ip interface g0/0** command on router R1.

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

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. The capitalized ACL names BROWSING and SURFING stand out in the screen output.

After an ACL configuration has been verified, the next step is to confirm that the ACL work as planned; blocking and permitting traffic as expected.

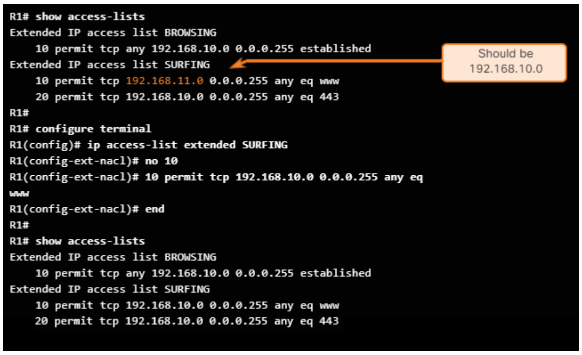
The guidelines discussed earlier in this section, suggest that ACL should be configured on a test network and then implemented on the production network.

Extended IPv4 ACL

Configure Extended IPv4 ACL

- 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.
    - ACE 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
```

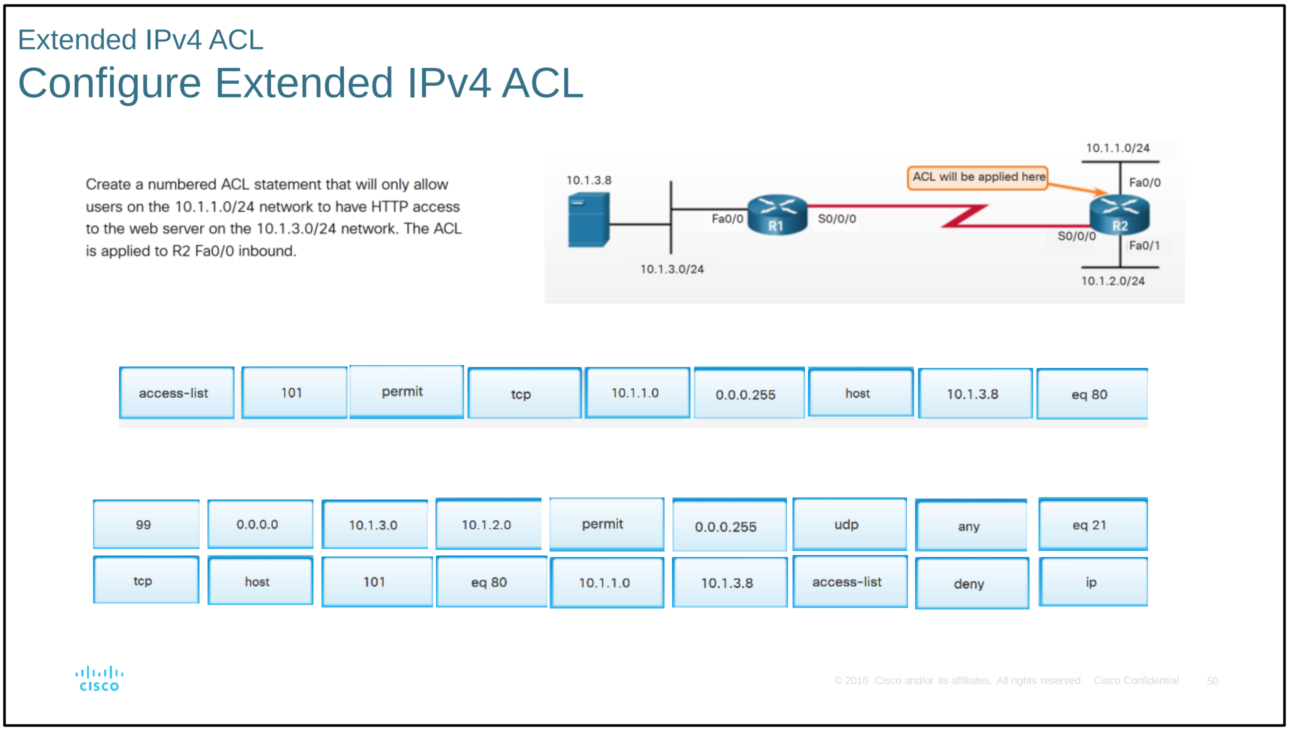
Editing Extended ACL

An extended ACL can be edited in one of two ways:

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 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. ACE can be inserted or removed.

In the figure 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.



Extended IPv4 ACL

Configure Extended IPv4 ACL

Create a numbered ACL statement that will only allow users on the 10.1.1.0/24 network to have HTTP access to the web server on the 10.1.3.0/24 network. The ACL is applied to R2 Fa0/0 inbound.

access-list	101	permit	tcp	10.1.1.0	0.0.0.255	host	10.1.3.8	eq 80

99	0.0.0.0	10.1.3.0	10.1.2.0	permit	0.0.0.255	udp	any	eq 21
tcp	host	101	eq 80	10.1.1.0	10.1.3.8	access-list	deny	ip

Editing Extended ACL

An extended ACL can be edited in one of two ways:

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 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. ACE can be inserted or removed.

In the figure 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.

IPv6 ACL

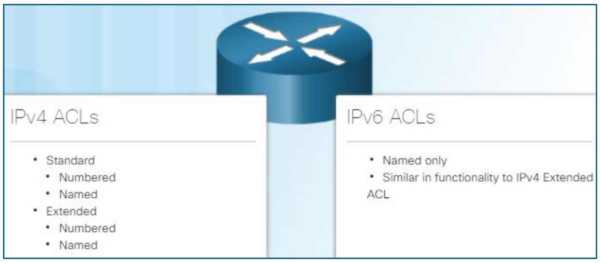
4 – Access Control Lists
4.3 – IPv6 ACL

IPv6 ACL

IPv6 ACL Creation

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

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



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 ACL in IPv6.

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

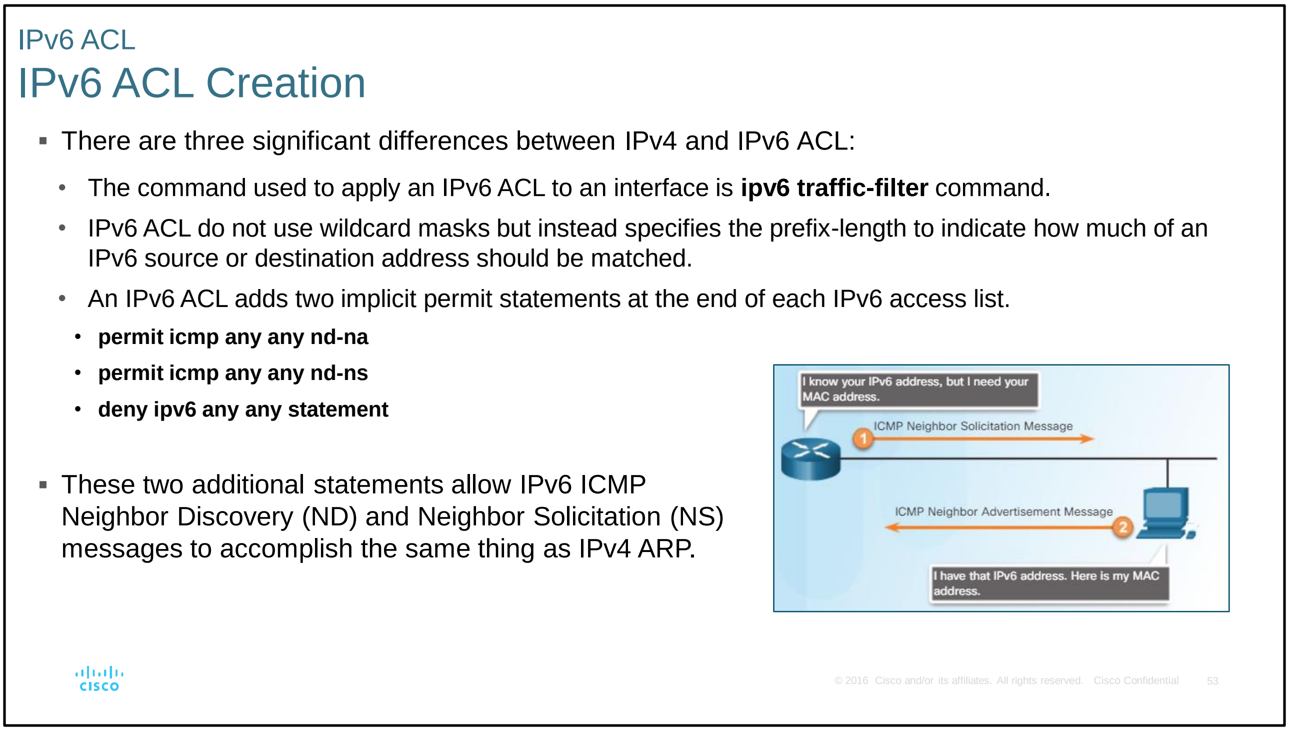
Types of IPv6 ACL

IPv6 ACL are similar to IPv4 ACL in both operation and configuration. Being familiar with IPv4 access lists makes IPv6 ACL easy to understand and configure.

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

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

An IPv4 ACL and an IPv6 ACL cannot share the same name.



IPv6 ACL

IPv6 ACL Creation

- There are three significant differences between IPv4 and IPv6 ACL:
  - The command used to apply an IPv6 ACL to an interface is **ipv6 traffic-filter** command.
  - IPv6 ACL 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.

Comparing IPv4 and IPv6 ACL

Although IPv4 and IPv6 ACL are similar, there are three significant differences between them:

The first difference is the command used to apply an IPv6 ACL to an interface. IPv4 uses the command **ip access-group** to apply an IPv4 ACL to an IPv4 interface. IPv6 uses the **ipv6 traffic-filter** command to perform the same function for IPv6 interfACE.

Unlike IPv4 ACL, IPv6 ACL 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.

The last major difference has to with the addition of two implicit permit statements at the end of each IPv6 access list. At the end of every IPv4 standard or extended ACL is an implicit **deny any** or **deny ip any any**. IPv6 includes a similar **deny ipv6 any any** statement at the end of each IPv6 ACL. The difference is IPv6 also includes two other implicit statements by default: **permit icmp any any nd-na** and **permit icmp any any nd-ns.**

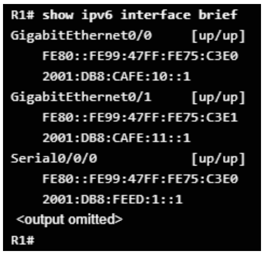
These two statements allow the router to participate in the IPv6 equivalent of ARP for IPv4. Recall that ARP is used in IPv4 to resolve Layer 3 addresses to Layer 2 MAC addresses. As shown in the figure, IPv6 uses ICMP Neighbor Discovery (ND) messages to accomplish the same thing. ND uses Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages.

ND messages are encapsulated in IPv6 packets and require the services of the IPv6 network layer while ARP for IPv4 does not use Layer 3. Because IPv6 uses the Layer 3 service for neighbor discovery, IPv6 ACL need to implicitly permit ND packets to be sent and received on an interface. Specifically, both Neighbor Discovery - Neighbor Advertisement (nd-na) and Neighbor Discovery - Neighbor Solicitation (nd-ns) messages are permitted.

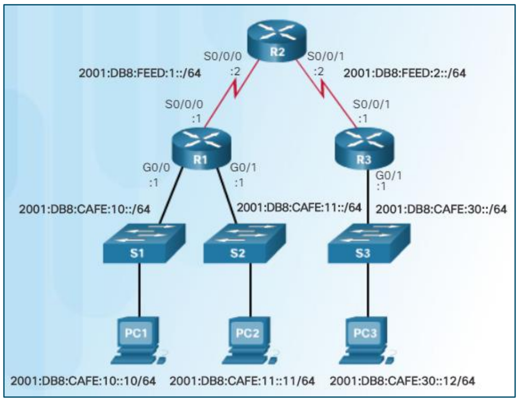
IPv6 ACL

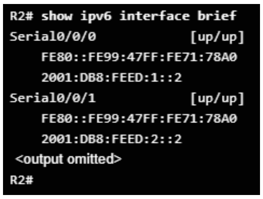
Configuring IPv6 ACL

- The following is the sample topology that will be used to demonstrate IPv6 ACL.
  - All interfACE are configured and active.

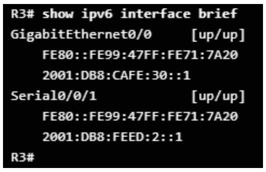


```
R1# show ipv6 interface brief
GigabitEthernet0/0      [up/up]
    FE80::FE99:47FF:FE75:C3E0
    2001:DB8:CAFE:10::1
GigabitEthernet0/1      [up/up]
    FE80::FE99:47FF:FE75:C3E1
    2001:DB8:CAFE:11::1
Serial0/0/0             [up/up]
    FE80::FE99:47FF:FE75:C3E0
    2001:DB8:FEED:1::1
 <output omitted>
R1#
```





```
R2# show ipv6 interface brief
Serial0/0/0             [up/up]
    FE80::FE99:47FF:FE71:78A0
    2001:DB8:FEED:1::2
Serial0/0/1             [up/up]
    FE80::FE99:47FF:FE71:78A0
    2001:DB8:FEED:2::2
 <output omitted>
R2#
```



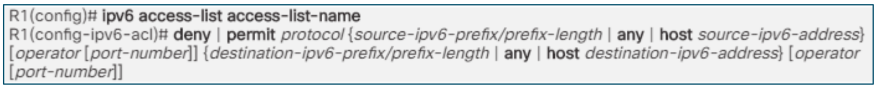
```
R3# show ipv6 interface brief
GigabitEthernet0/0      [up/up]
    FE80::FE99:47FF:FE71:7A20
    2001:DB8:CAFE:30::1
Serial0/0/1             [up/up]
    FE80::FE99:47FF:FE71:7A20
    2001:DB8:FEED:2::1
R3#
```

Configuring IPv6 Topology
Figure 1 shows the topology that will be used for configuring IPv6 ACL. The topology is similar to the previous IPv4 topology except for the IPv6 addressing scheme. There are three 2001:DB8:CAFE::/64 subnets:
2001:DB8:CAFE:10::/64
2001:DB8:CAFE:11::/64
2001:DB8:CAFE:30::/64
Two serial networks connect the three routers:
2001:DB8:FEED:1::/64
2001:DB8:FEED:2::/64
Figures 2, 3, and 4 show the IPv6 address configuration for each router. The **show ipv6 interface brief** command is used to verify the address and the state of the interface.
Note: The **no shutdown** command and the **clock rate** command are not shown.

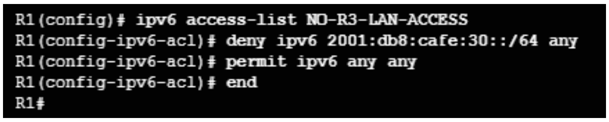
IPv6 ACL

Configuring IPv6 ACL

- In IPv6 there are only named ACL and the configuration is similar to IPv4 extended named ACL.

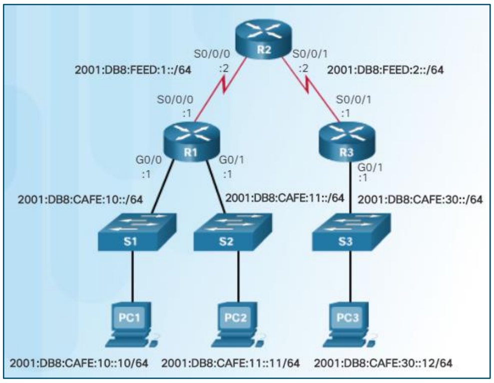


```
R1(config)# ipv6 access-list access-list-name
R1(config-ipv6-acl)# deny | permit protocol {source-ipv6-prefix/prefix-length | any | host source-ipv6-address}
[operator [port-number]] {destination-ipv6-prefix/prefix-length | any | host destination-ipv6-address} [operator
[port-number]]
```



```
R1(config)# ipv6 access-list NO-R3-LAN-ACCESS
R1(config-ipv6-acl)# deny ipv6 2001:db8:cafe:30::/64 any
R1(config-ipv6-acl)# permit ipv6 any any
R1(config-ipv6-acl)# end
R1#
```

- In this example:
  - The 1st statement names the IPv6 ACL **NO-R3-LAN-ACCESS**.
  - The 2nd statement denies all IPv6 packets from the 2001:DB8:CAFE:30::/64 destined for any IPv6 network.
  - The 3rd statement allows all other IPv6 packets.



Configuring IPv6 ACL

In IPv6 there are only named ACL. The configuration is similar to that of an IPv4 extended named ACL.

Figure 1 shows the command syntax for IPv6 ACL. The syntax is similar to the syntax used for an IPv4 extended ACL. One significant difference is the use of the IPv6 prefix-length instead of an IPv4 wildcard mask.

There are three basic steps to configure an IPv6 ACL:

Step 1. From global configuration mode, use the **ipv6 access-list** *name* command to create an IPv6 ACL. Like IPv4 named ACL, IPv6 names are alphanumeric, case sensitive, and must be unique. Unlike IPv4, there is no need for a standard or extended option.

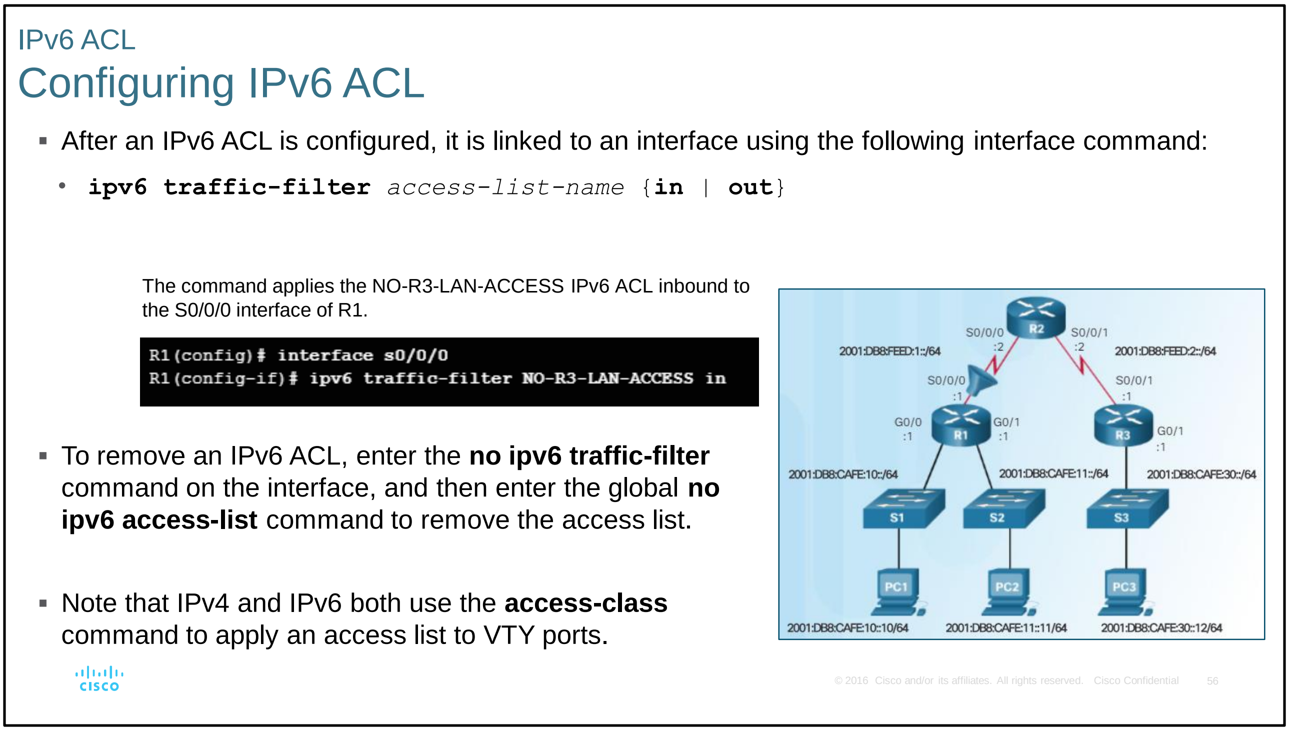
Step 2. From the named ACL configuration mode, use the **permit** or **deny** statements to specify one or more conditions to determine if a packet is forwarded or dropped.

Step 3. Return to privileged EXEC mode with the **end** command.

Figure 2 demonstrates the steps to create an IPv6 ACL with a simple example based on the previous topology. The first statement names the IPv6 access list NO-R3-LAN-ACCESS. Similar to IPv4 named ACL, capitalizing IPv6 ACL names is not required, but makes them stand out when viewing the running-config output.

The second statement denies all IPv6 packets from the 2001:DB8:CAFE:30::/64 destined for any IPv6 network. The third statement allows all other IPv6 packets.

Figure 3 shows the ACL in context with the topology.



IPv6 ACL

Configuring IPv6 ACL

- After an IPv6 ACL is configured, it is linked to an interface using the following interface command:
  - **ipv6 traffic-filter** *access-list-name* {**in** | **out**}

The command applies the NO-R3-LAN-ACCESS IPv6 ACL inbound to the S0/0/0 interface of R1.

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

- To remove an IPv6 ACL, enter the **no ipv6 traffic-filter** command on the interface, and then enter the global **no ipv6 access-list** command to remove the access list.
- Note that IPv4 and IPv6 both use the **access-class** command to apply an access list to VTY ports.

Applying an IPv6 ACL to an Interface

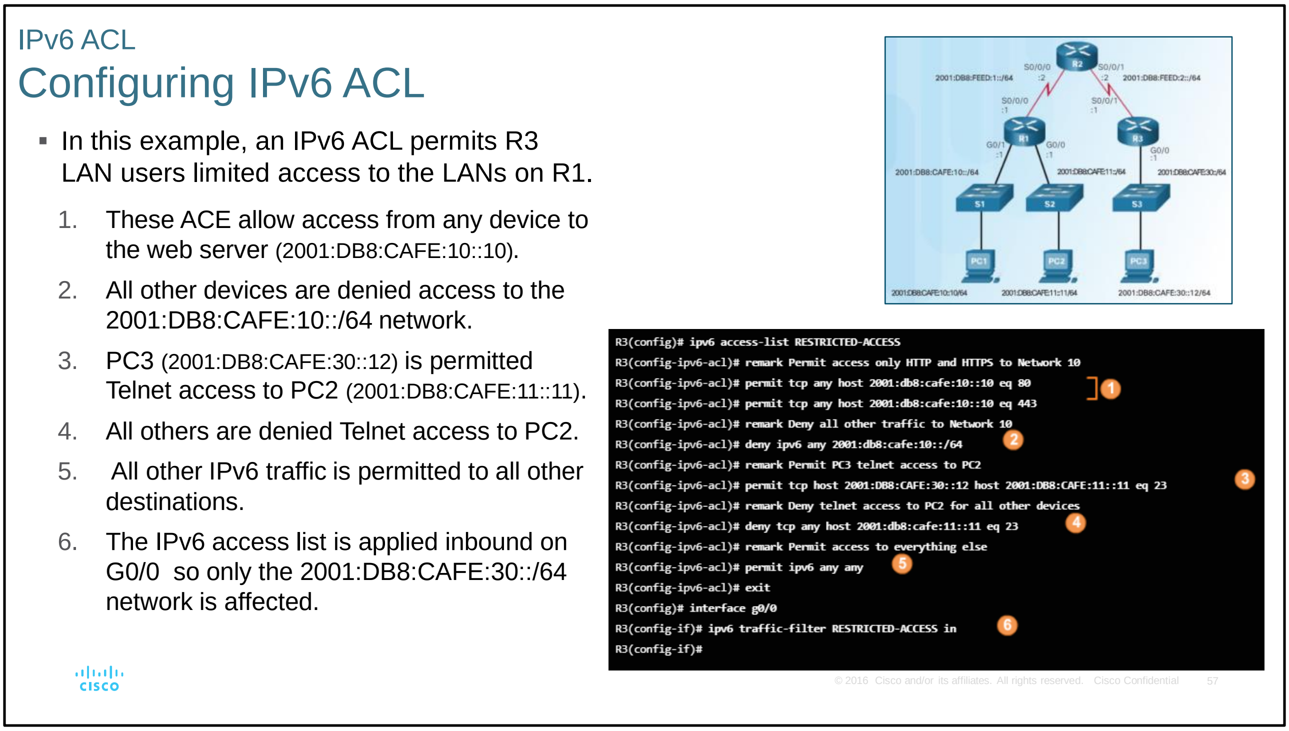
After an IPv6 ACL is configured, it is linked to an interface using the **ipv6 traffic-filter** command:

Router(config-if)# **ipv6 traffic-filter** *access-list-name* { **in** | **out** }

The figure shows the NO-R3-LAN-ACCESS ACL configured previously and the commands used to apply the IPv6 ACL inbound to the S0/0/0 interface. Applying the ACL to the inbound S0/0/0 interface will deny packets from 2001:DB8:CAFE:30::/64 to both of the LANs on R1.

To remove an ACL from an interface, first enter the **no ipv6 traffic-filter** command on the interface, and then enter the global **no ipv6 access-list** command to remove the access list.

Note: IPv4 and IPv6 both use the **access-class** command to apply an access list to VTY ports.



IPv6 ACL

Configuring IPv6 ACL

- In this example, an IPv6 ACL permits R3 LAN users limited access to the LANs on R1.
  1. These ACE allow access from any device to the web server (2001:DB8:CAFE:10::10).
  2. All other devices are denied access to the 2001:DB8:CAFE:10::/64 network.
  3. PC3 (2001:DB8:CAFE:30::12) is permitted Telnet access to PC2 (2001:DB8:CAFE:11::11).
  4. All others are denied Telnet access to PC2.
  5. All other IPv6 traffic is permitted to all other destinations.
  6. The IPv6 access list is applied inbound on G0/0 so only the 2001:DB8:CAFE:30::/64 network is affected.

```
R3(config)# ipv6 access-list RESTRICTED-ACCESS
R3(config-ipv6-acl)# remark Permit access only HTTP and HTTPS to Network 10
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
R3(config-ipv6-acl)# remark Deny all other traffic to Network 10
R3(config-ipv6-acl)# deny ipv6 any 2001:db8:cafe:10::/64
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:CAFE:11::11 eq 23
R3(config-ipv6-acl)# remark Deny telnet access to PC2 for all other devices
R3(config-ipv6-acl)# deny tcp any host 2001:db8:cafe:11::11 eq 23
R3(config-ipv6-acl)# remark Permit access to everything else
R3(config-ipv6-acl)# permit ipv6 any any
R3(config-ipv6-acl)# exit
R3(config)# interface g0/0
R3(config-if)# ipv6 traffic-filter RESTRICTED-ACCESS in
R3(config-if)#
```

57

IPv6 ACL Example

Deny FTP

The topology for the examples is shown in Figure 1.

Restricted Access

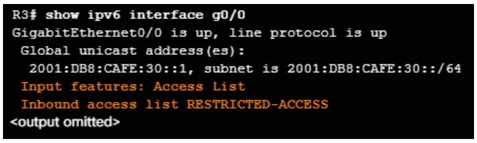
In the example (Figure 2), an IPv6 ACL is configured to give the LAN on R3 limited access to the LANs on R1. Comments are added in the configuration to document the ACL. The following features have been labelled in the ACL:

1. The first two permit statements allow access from any device to the web server at 2001:DB8:CAFE:10::10.
2. All other devices are denied access to the 2001:DB8:CAFE:10::/64 network.
3. PC3 at 2001:DB8:CAFE:30::12 is permitted Telnet access to PC2 which has the IPv6 address 2001:DB8:CAFE:11::11.
4. All other devices are denied Telnet access to PC2.
5. All other IPv6 traffic is permitted to all other destinations.
6. The IPv6 access list is applied to interface G0/0 in the inbound direction, so only the 2001:DB8:CAFE:30::/64 network is affected.

IPv6 ACL

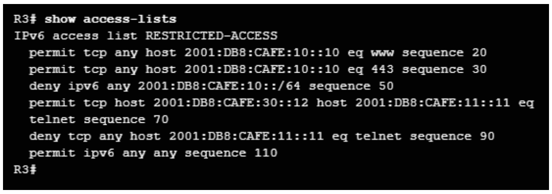
Configuring IPv6 ACL

- The commands used to verify an IPv6 access list are similar to those used for IPv4 ACL.
- Use the **show ipv6 interface** command to see which ACL and direction is configured on an interface.



```
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
<output omitted>
```

- Use the **show access-lists** command displays all configured IPv4 and IPv6 access lists
  - Notice that IPv6 ACL sequence numbers are displayed at the end of the ACE.



```
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#
```

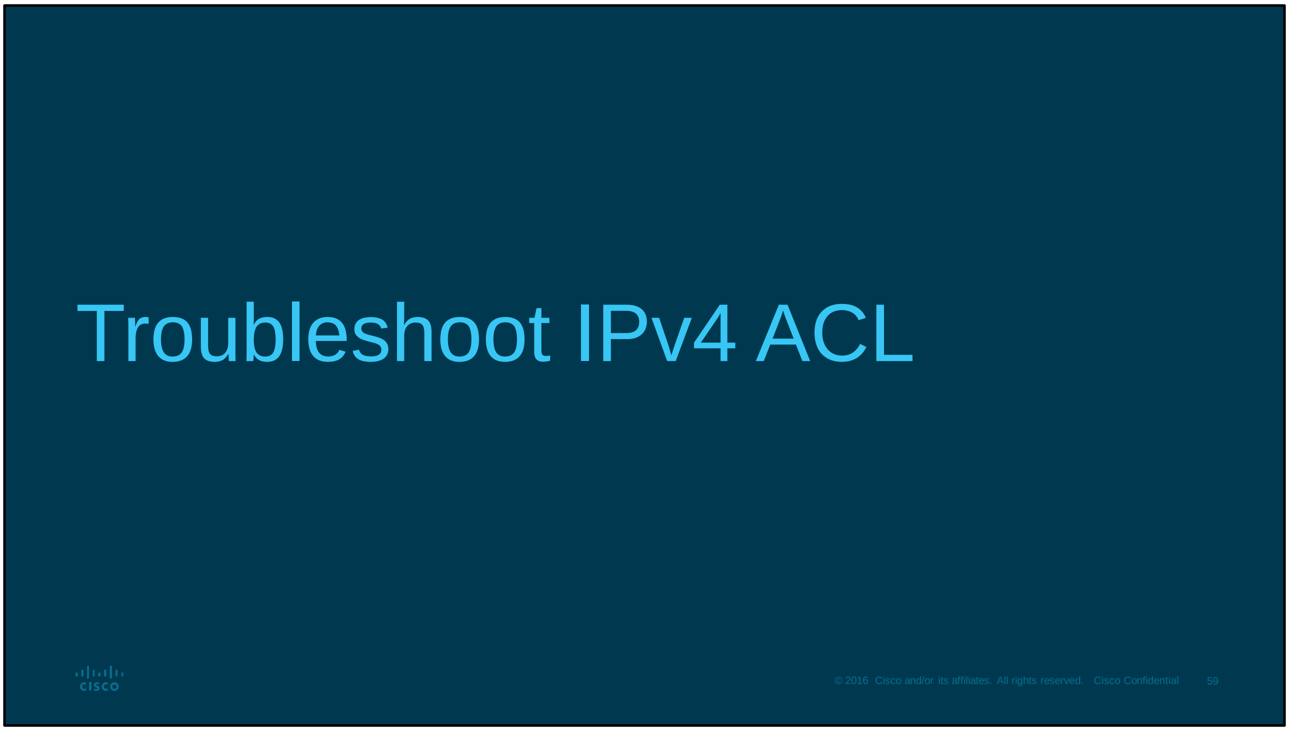
- The **show running-config** command displays all of the ACE and remark statements.

Verifying IPv6 ACL

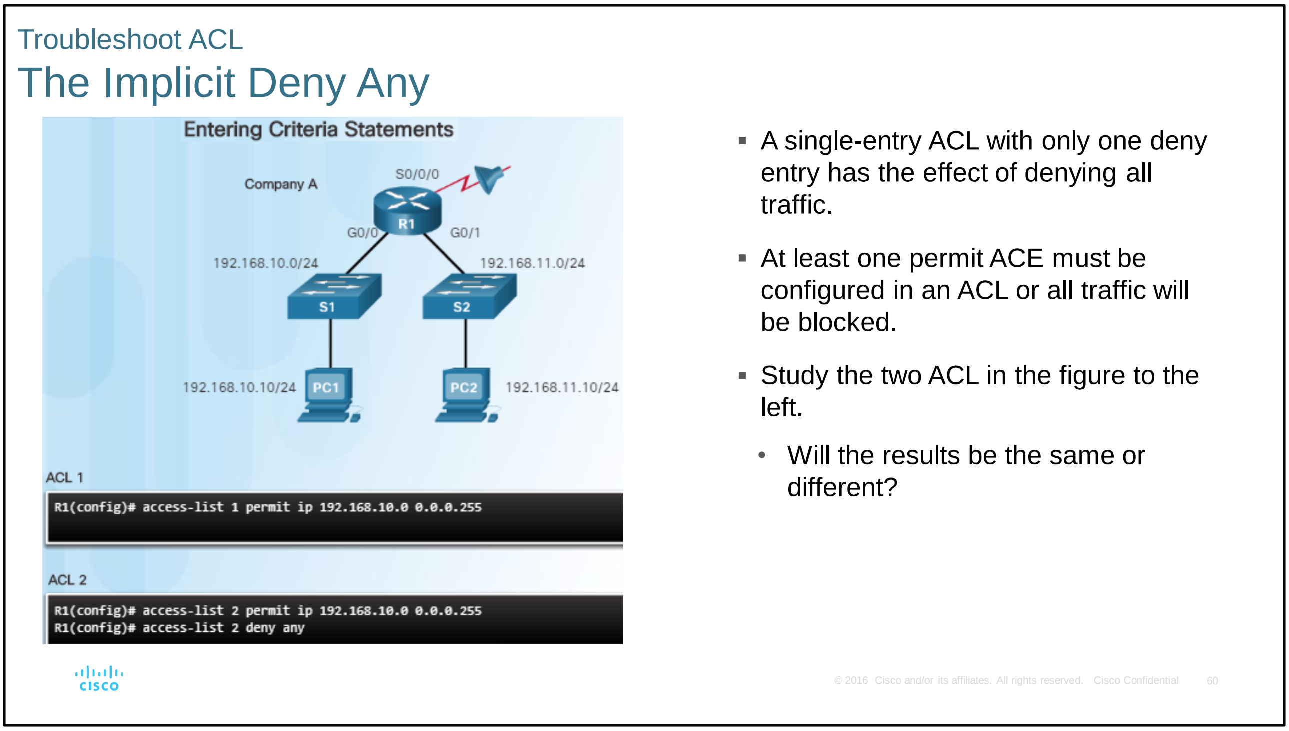
The commands used to verify an IPv6 access list are similar to those used for IPv4 ACL. Using these commands, the IPv6 access list RESTRICTED-ACCESS that was configured previously can be verified. Figure 1 shows the output of the **show ipv6 interface** command. The output confirms that RESTRICTED-ACCESS ACL is configured inbound on the G0/0 interface.

As shown in Figure 2, the **show access-lists** command displays all access lists on the router including both IPv4 and IPv6 ACL. Notice that with IPv6 ACL the sequence numbers occur at the end of the statement and not the beginning as with IPv4 access lists. Although the statements appear in the order they were entered, they are not always incremented by 10. This is because the remark statements that were entered use a sequence number but are not displayed in the output of the **show access-lists** command.

Similar to extended ACL for IPv4, IPv6 access lists are displayed and processed in the order the statements are entered. Remember, IPv4 standard ACL use an internal logic which changes their order and processing sequence.



Troubleshoot IPv4 ACL



Troubleshoot ACL

The Implicit Deny Any

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
```

- A single-entry ACL with only one deny entry has the effect of denying all traffic.
- At least one permit ACE must be configured in an ACL or all traffic will be blocked.
- Study the two ACL in the figure to the left.
  - Will the results be the same or different?

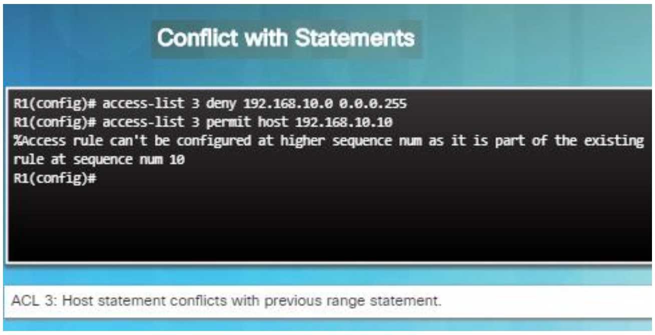
The Implicit Deny Any

A single-entry ACL with only one deny entry has the effect of denying all traffic. At least one permit ACE must be configured in an ACL or all traffic is blocked.

For the network in the figure, applying either ACL 1 or ACL 2 to the S0/0/0 interface of R1 in the outbound direction will have the same effect. Network 192.168.10.0 will be permitted to access the networks reachable through S0/0/0, while 192.168.11.0 will not be allowed to access those networks. In ACL 1, if a packet does not match the permit statement, it is discarded.

Troubleshoot ACL

The Order of ACE in an ACL



Conflict with Statements

```
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.

- The order in which ACE are configured are important since ACE are processed sequentially.
- The figure to the left demonstrates a conflict between two statements since they are in the wrong order.
  - The first deny statement blocks everything in the 192.168.10.0/24 network.
  - However, the second permit statement is attempting to allow host 192.168.10.10 through.
  - This statement is rejected since it is a subset of the previous statement.
  - Reversing the order of these two statements will solve the problem.

The Order of ACE in an ACL

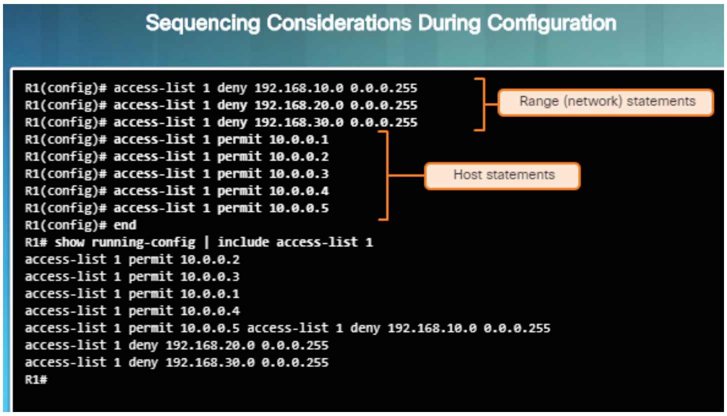
Cisco IOS applies an internal logic when accepting and processing standard ACE. As discussed previously, ACE are processed sequentially; therefore, the order in which ACE are entered is important.

For example, in Figure, ACL 3 contains two ACE. The first ACE uses a wildcard mask to deny a range of addresses, which includes all hosts in the 192.168.10.0/24 network. The second ACE is a host statement that examines a specific host, 192.168.10.10, that belongs to the 192.168.10.0/24 network. The IOS internal logic for standard access lists rejects the second statement and returns an error message because it is a subset of the previous statement.

Reversing the order of these two statements will solve the problem. This is a valid sequence of statements because the first statement refers a specific host, not a range of hosts.

Troubleshoot ACL

Cisco IOS Reorders Standard ACL


Sequencing Considerations During Configuration

```
R1(config)# access-list 1 deny 192.168.10.0 0.0.0.255
R1(config)# access-list 1 deny 192.168.20.0 0.0.0.255
R1(config)# access-list 1 deny 192.168.30.0 0.0.0.255
R1(config)# access-list 1 permit 10.0.0.1
R1(config)# access-list 1 permit 10.0.0.2
R1(config)# access-list 1 permit 10.0.0.3
R1(config)# access-list 1 permit 10.0.0.4
R1(config)# access-list 1 permit 10.0.0.5
R1(config)# end
R1# show running-config | include access-list 1
access-list 1 permit 10.0.0.2
access-list 1 permit 10.0.0.3
access-list 1 permit 10.0.0.1
access-list 1 permit 10.0.0.4
access-list 1 permit 10.0.0.5 access-list 1 deny 192.168.10.0 0.0.0.255
access-list 1 deny 192.168.20.0 0.0.0.255
access-list 1 deny 192.168.30.0 0.0.0.255
R1#
```

- Note the order in which the access-list statements were entered during configuration.
- Notice how the order was changed when you enter the **show running-config** command.
- The host statements are listed first, however, not in the order 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.
- The range statements are displayed in the order they were entered. The hashing function is applied to host statements.

Cisco IOS Reorders Standard ACL

The order in which standard ACE are entered may not be the order that they are stored, displayed, or processed by the router.

Figure shows the configuration of a standard access list. Range statements that deny three networks are configured first followed by five host statements. The host statements are all valid statements because their host IPv4 addresses are not part of the previously entered range statements.

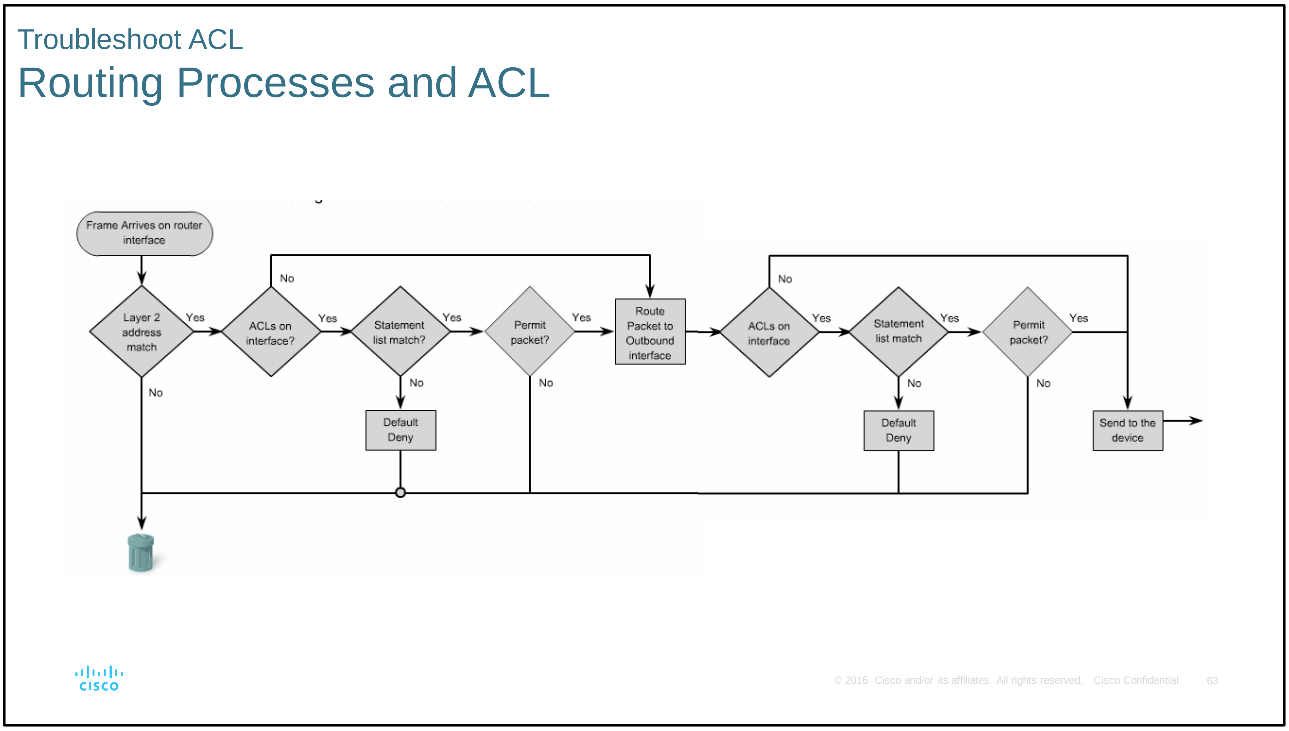
The **show running-config** command is used to verify the ACL configuration. Notice that the statements are listed in a different order than they were entered. We will use the **show access-lists** command to understand the logic behind this.

As shown in Figure, the **show access-lists** command displays ACE along with their sequence numbers. We might expect the order of the statements in the output to reflect the order in which they were entered. However, the **show access-lists** output shows that this is not the case.

The order in which the standard ACE are listed is the sequence used by the IOS to process the list. Notice that the statements are grouped into two sections, host statements followed by range statements. The sequence number indicates the order that the statement was entered, not the order the statement will be processed.

The host statements are listed first 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. The range statements are displayed after the host statements. These statements are listed in the order in

which they were entered.
Note: The hashing function is only applied to host statements in an IPv4 standard access list. The details of the hashing function are beyond the scope of this course.
Recall that standard and numbered ACL can be edited using sequence numbers. When inserting a new ACL statement, the sequence number will only affect the location of a range statement in the list. Host statements will always be put in order using the hashing function.
Continuing with the example, after saving the running-configuration, the router is reloaded. As shown in Figure 2, the **show access-lists** command displays the ACL in the same order, however the statements have been renumbered. The sequence numbers are now in numerical order.



Troubleshoot ACL

Routing Processes and ACL

Frame Arrives on router interface

Layer 2 address match

Yes

No

ACLs on interface?

No

Yes

Statement list match?

Yes

No

Default Deny

Permit packet?

Yes

No

Route Packet to Outbound interface

ACLs on interface

No

Yes

Statement list match

Yes

No

Default Deny

Permit packet?

Yes

No

Send to the device

Routing Processes and ACL

The figure shows the logic of routing and ACL processes. When a packet arrives at a router interface, the router process is the same, whether ACL are used or not. As a frame enters an interface, the router checks to see whether the destination Layer 2 address matches its interface Layer 2 address, or whether 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.

If the packet matches a statement, the packet is either permitted or denied. 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.

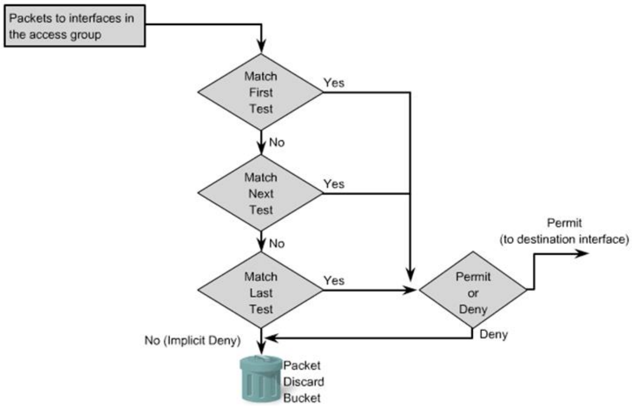
If the packet matches a statement, it is either permitted or denied.

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 ACL

Processing Packets with ACL

- It is beneficial to consider how an inbound and outbound ACL is processed.
- Inbound ACL 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.



Packets to interfaces in the access group

Match First Test

Yes

No

Match Next Test

Yes

No

Match Last Test

Yes

No (Implicit Deny)

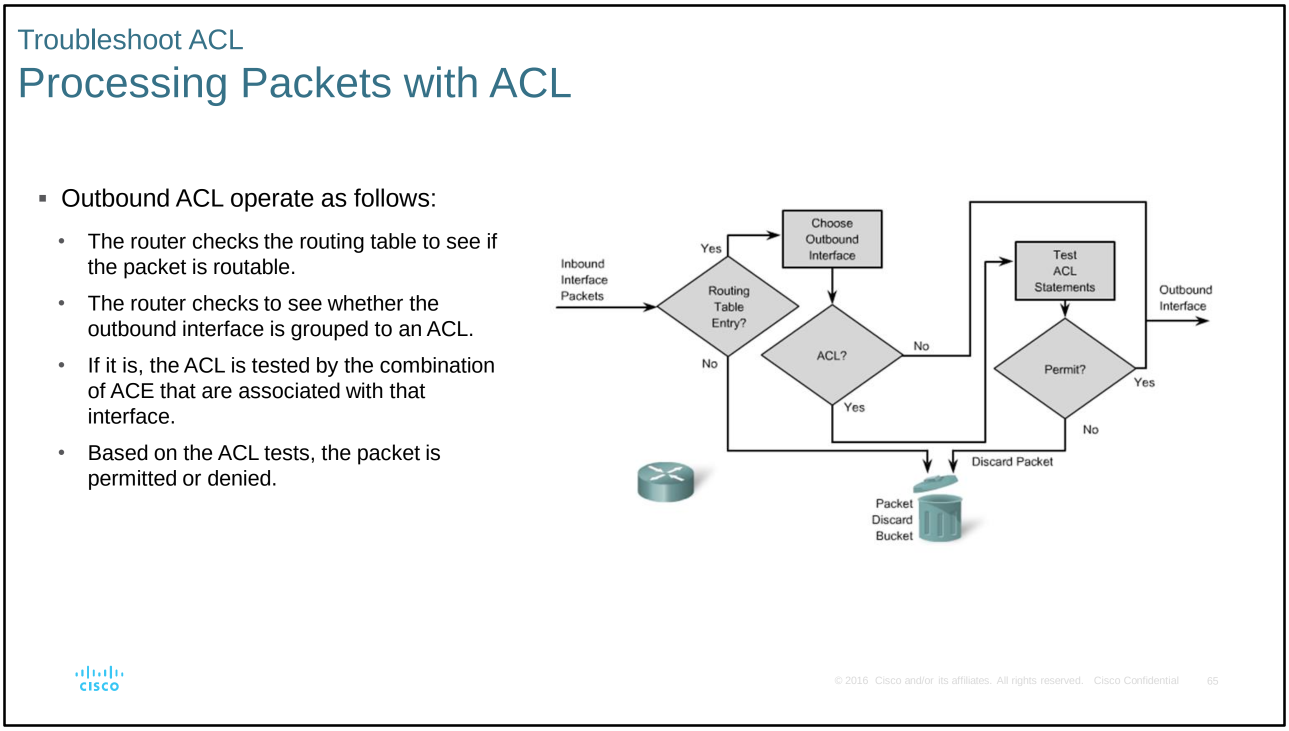
Permit or Deny

Permit (to destination interface)

Deny

Packet Discard Bucket

4.4 – Troubleshoot ACL
4.4.1 – Processing Packets with ACL
4.4.1.1 – Inbound and Outbound ACL Logic

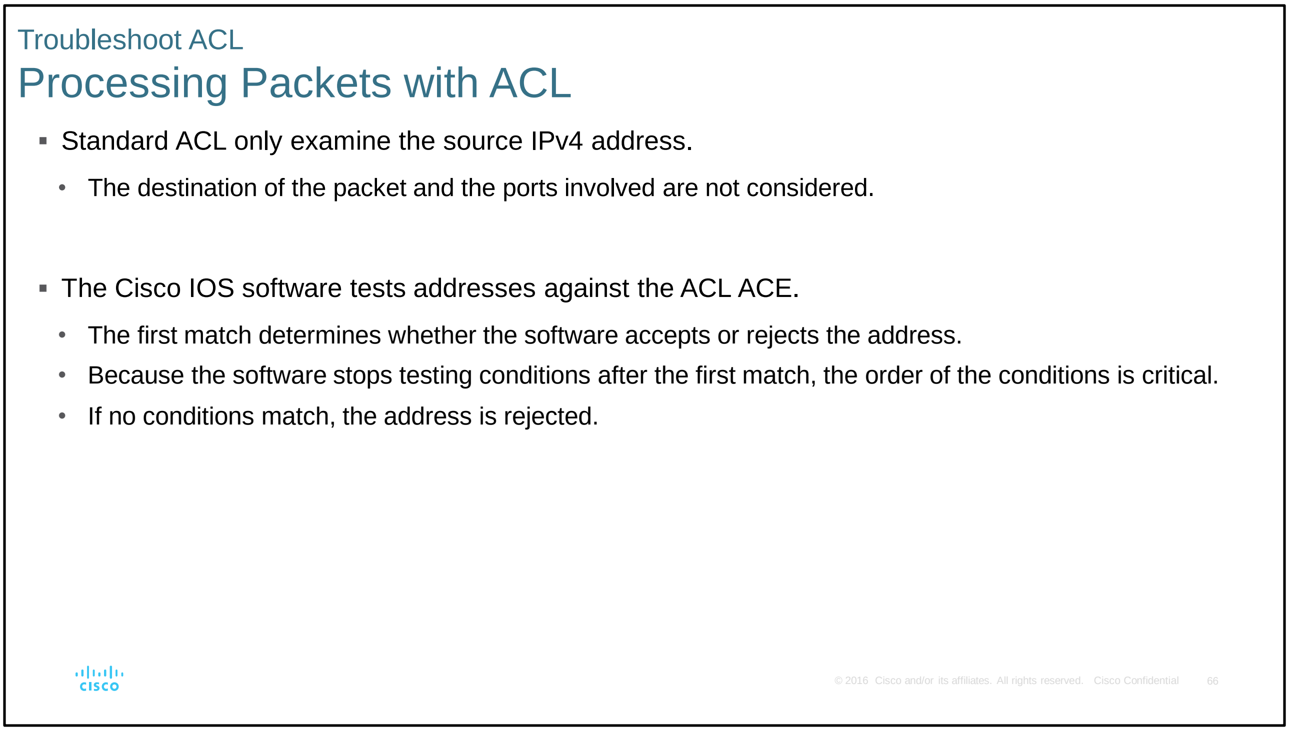


Troubleshoot ACL

Processing Packets with ACL

- Outbound ACL 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 ACE that are associated with that interface.
  - Based on the ACL tests, the packet is permitted or denied.

4.4 – Troubleshoot ACL
4.4.1 – Processing Packets with ACL
4.4.1.1 – Inbound and Outbound ACL Logic

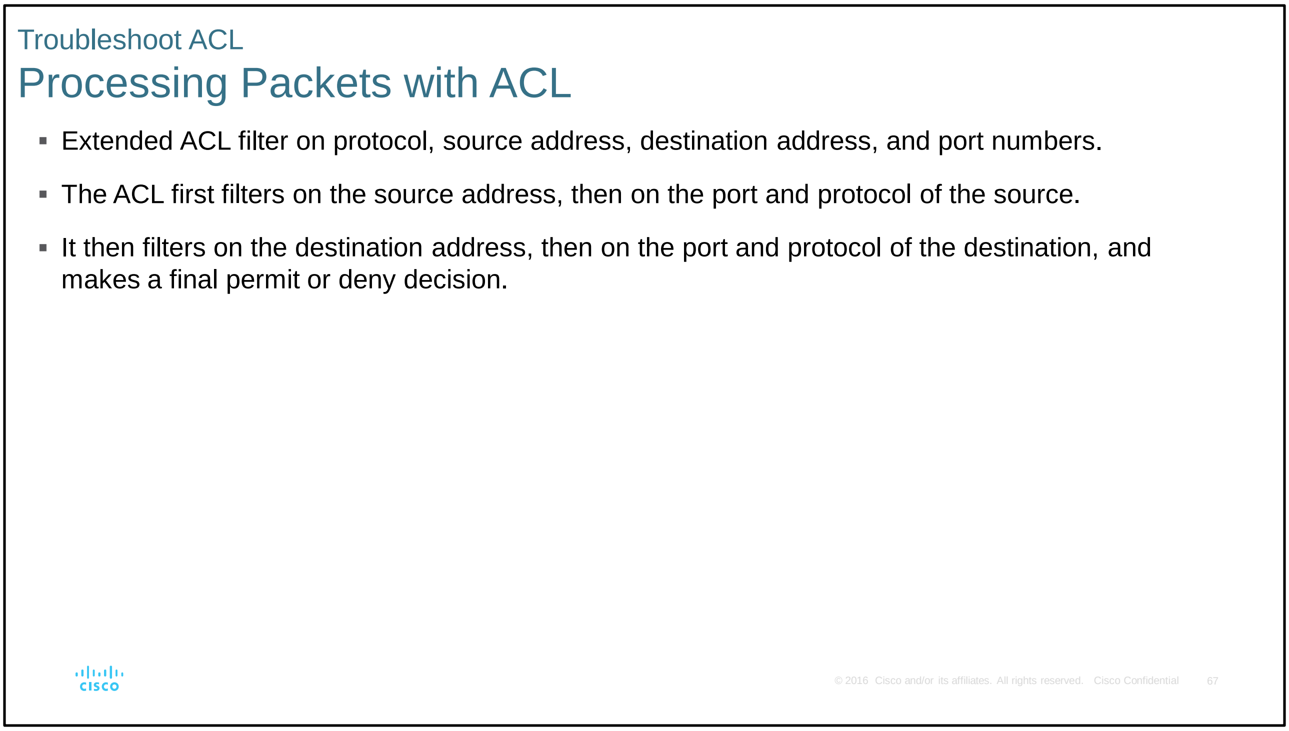


Troubleshoot ACL

Processing Packets with ACL

- Standard ACL 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 ACE.
  - 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.

4.4 – Troubleshoot ACL
4.4.1 – Processing Packets with ACL
4.4.1.3 – Standard ACL Decision Process



Troubleshoot ACL

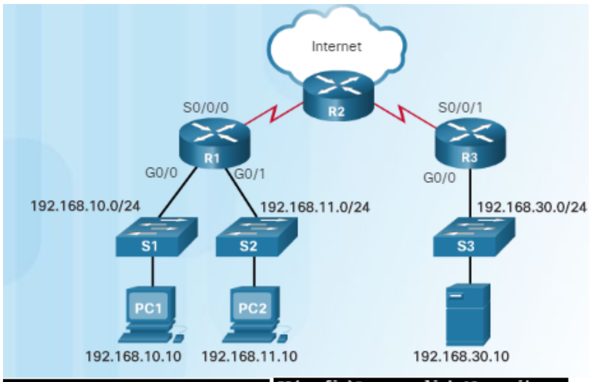
Processing Packets with ACL

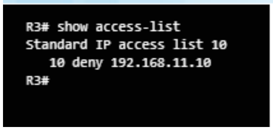
- Extended ACL filter on protocol, source address, destination address, and port numbers.
- The ACL first filters on the source address, then on the port and protocol of the source.
- It then filters on the destination address, then on the port and protocol of the destination, and makes a final permit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

4.4 – Troubleshoot ACL
4.4.1 – Processing Packets with ACL
4.4.1.4 – Extended ACL Decision Process

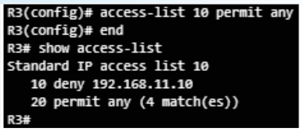
Common IPv4 ACL Errors

Troubleshooting Standard IPv4 ACL – Example 1





```
R3# show access-list
Standard IP access list 10
   10 deny 192.168.11.10
R3#
```



```
R3(config)# access-list 10 permit any
R3(config)# end
R3# show access-list
Standard IP access list 10
   10 deny 192.168.11.10
   20 permit any (4 match(es))
R3#
```

- The most common errors involving ACL:
  - Entering ACE in the wrong order
  - Not specifying adequate ACL rules
  - Applying the ACL using the wrong direction, wrong interface, or wrong source address
- In the figure to the left, PC2 should not be able to access the File Server. However, PC1 can not access it either.
- The output of the show access-list command shows the one deny statement in the ACL.
- The set of commands on the right shows the solution. The permit statement allows other devices to access since the implicit deny was blocking other traffic.

Troubleshooting Standard IPv4 ACL - Example 1

Using the **show** commands described earlier reveals most of the more common ACL errors. The most common errors are entering ACE in the wrong order and not specifying adequate ACL rules. Other common errors include applying the ACL using the wrong direction, the wrong interface, or the wrong source addresses.

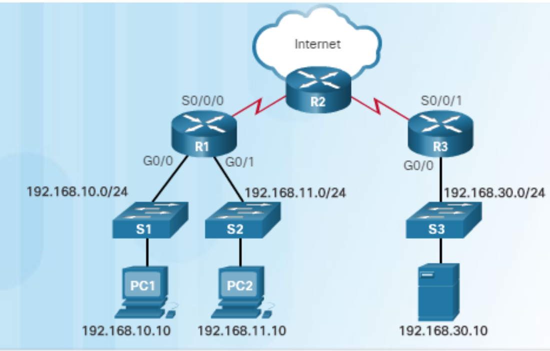
Security Policy: PC2 should not be able to access the File Server.

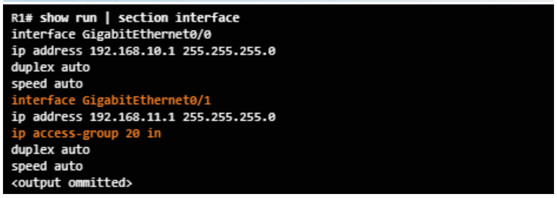
In Figure 1, although PC2 cannot access the File Server, neither can PC1. When viewing the output of the **show access-list** command, only PC2 is explicitly denied. However, there is no permit statement allowing other access.

Solution: All access out the G0/0 interface to the 192.168.30.0/24 LAN is currently implicitly denied. Add a statement to ACL 10 to permit all other traffic, as shown in Figure 2. PC1 should now be able to access the file server. Output from the **show access-list** command verifies that a ping from PC1 to the File Server matches the permit any statement.

Common IPv4 ACL Errors

Troubleshooting Standard IPv4 ACL – Example 2





```
R1# show run | section interface
interface GigabitEthernet0/0
ip address 192.168.10.1 255.255.255.0
duplex auto
speed auto
interface GigabitEthernet0/1
ip address 192.168.11.1 255.255.255.0
ip access-group 20 in
duplex auto
speed auto
<output ommitted>
```

- The 192.168.11.0/24 network should not be able to access the 192.168.10.0/24 network.
- PC2 cannot access PC1 as planned, however, it also cannot access the Internet through R2.
- Problem: access-list 20 was applied to G0/1 on an inbound direction
- Where should ACL 20 be applied and in which direction?
- In order for PC2 to access the Internet, ACL 20 needs to be removed from the G0/1 interface and applied outbound on the G0/0 interface.

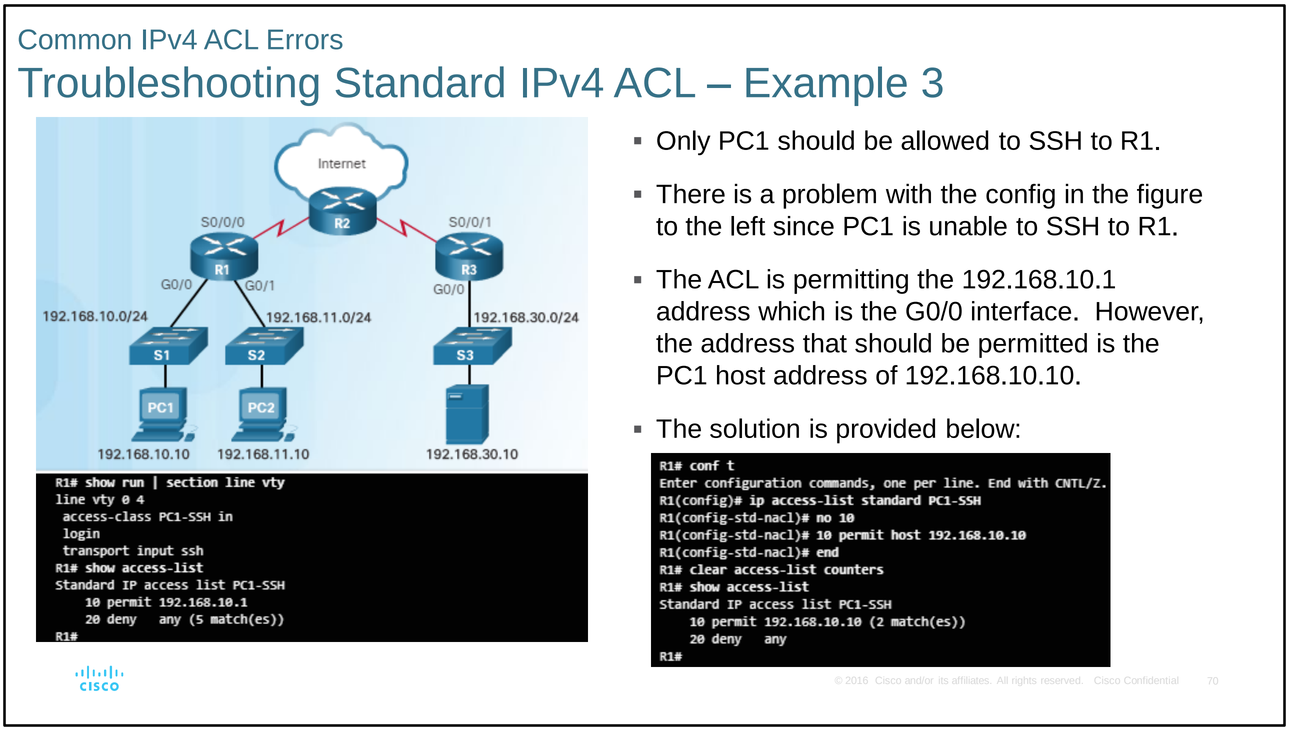
Troubleshooting Standard IPv4 ACL - Example 2

Security Policy: The 192.168.11.0/24 network should not be able to access the 192.168.10.0/24 network.

In Figure 1, PC2 cannot access PC1. Nor can it access the Internet through R2. When viewing the output of the **show access-list** command, you can see that PC2 is matching the deny statement. ACL 20 seems to be configured correctly. You suspect that it must be incorrectly applied and view the interface configurations for R1

In Figure 2, the **show run** command filtered to view the interface configurations reveals that ACL 20 was applied to the wrong interface and in the wrong direction. All traffic from the 192.168.11.0/24 is denied inbound access through the G0/1 interface.

Solution: To correct this error, remove ACL 20 from the G0/1 interface and apply it outbound on the G0/0 interface, as shown in Figure 3. PC2 cannot access PC1 but can now access the Internet.



Common IPv4 ACL Errors

Troubleshooting Standard IPv4 ACL – Example 3

```
R1# show run | section line vty
line vty 0 4
 access-class PC1-SSH in
 login
 transport input ssh
R1# show access-list
Standard IP access list PC1-SSH
    10 permit 192.168.10.1
    20 deny   any (5 match(es))
R1#
```

- Only PC1 should be allowed to SSH to R1.
- There is a problem with the config in the figure to the left since PC1 is unable to SSH to R1.
- The ACL is permitting the 192.168.10.1 address which is the G0/0 interface. However, the address that should be permitted is the PC1 host address of 192.168.10.10.
- The solution is provided below:

```
R1# conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# ip access-list standard PC1-SSH
R1(config-std-nacl)# no 10
R1(config-std-nacl)# 10 permit host 192.168.10.10
R1(config-std-nacl)# end
R1# clear access-list counters
R1# show access-list
Standard IP access list PC1-SSH
    10 permit 192.168.10.10 (2 match(es))
    20 deny   any
R1#
```

Troubleshooting Standard IPv4 ACL - Example 3

Security Policy: Only PC1 is allowed SSH remote access to R1.

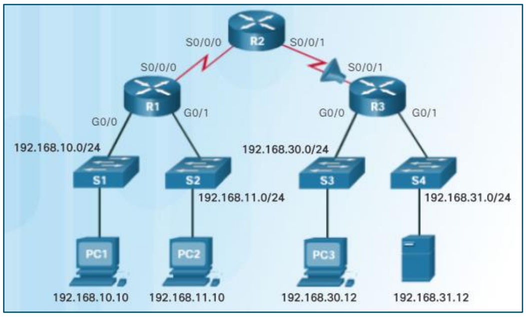
In Figure 1, PC1 is unable to remotely access R1 using an SSH connection. Viewing the running configuration section for the VTY lines reveals that an ACL named PC1-SSH is correctly applied for inbound connections. The VTY lines are correctly configured to only allow SSH connections. From the output of the show access-list command, you notice that the IPv4 address is the G0/0 interface for R1, not the IPv4 address of PC1. Also, notice that the administrator configured an explicit deny any statement in the ACL. This is helpful because, in this situation, you will see matches for failed attempts to remotely access R1.

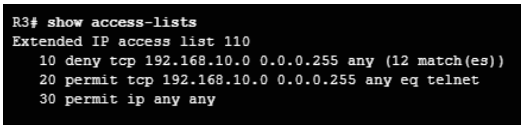
Solution: Figure 2 shows the process for correcting the error. Because the statement that needs to be corrected is the first statement, we use the sequence number 10 to delete it by entering **no 10**. We then configure the correct IPv4 address for PC1. The **clear access-list counters** command resets the output to only show new matches. An attempt from PC2 to remotely access R1 is successful, as shown in the output for the **show access-list** command.

Common IPv4 ACL Errors

Troubleshooting Extended IPv4 ACL – Example 1

- 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
```

Troubleshooting IPv4 ACL - Example 1

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

In the figure, host 192.168.10.10 has no Telnet 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.

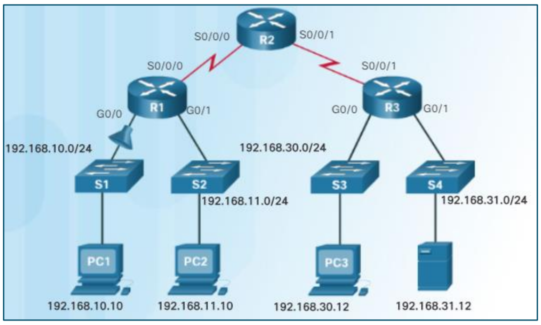
Solution - Look at the order of the ACE. Host 192.168.10.10 has no connectivity with 192.168.30.12 because of the order of rule 10 in the access list. Because the router processes ACL from the top down, statement 10 denies host 192.168.10.10, so statement 20 can never be matched. Statements 10 and 20 should be reversed. The last line allows all other non-TCP traffic that falls under IP (ICMP, UDP, etc.).

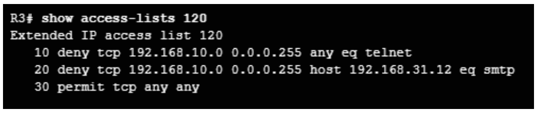
Common IPv4 ACL Errors

Troubleshooting Extended IPv4 ACL – Example 2

- 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
```

Troubleshooting IPv4 ACL - Example 2

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

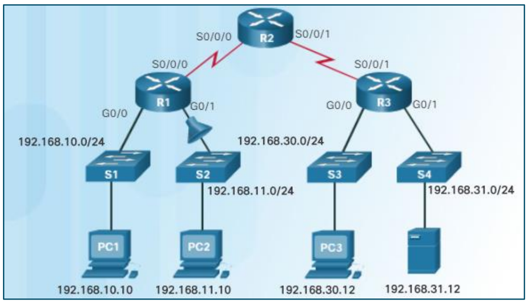
Solution - The 192.168.10.0/24 network cannot use TFTP to connect to the 192.168.30.0/24 network because TFTP uses the transport protocol UDP. Statement 30 in access list 120 allows all other TCP traffic. However, because TFTP uses UDP instead of TCP, it is implicitly denied. Recall that the implied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

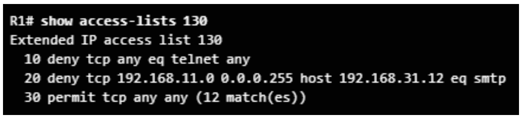
This ACL works whether it is applied to G0/0 of R1, or S0/0/1 of R3, or S0/0/0 of R2 in the incoming direction. However, based on the rule about placing extended ACL closest to the source, the best option is to place it inbound on G0/0 of R1 because it allows undesirable traffic to be filtered without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

Common IPv4 ACL Errors

Troubleshooting Extended IPv4 ACL – Example 3

- 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.





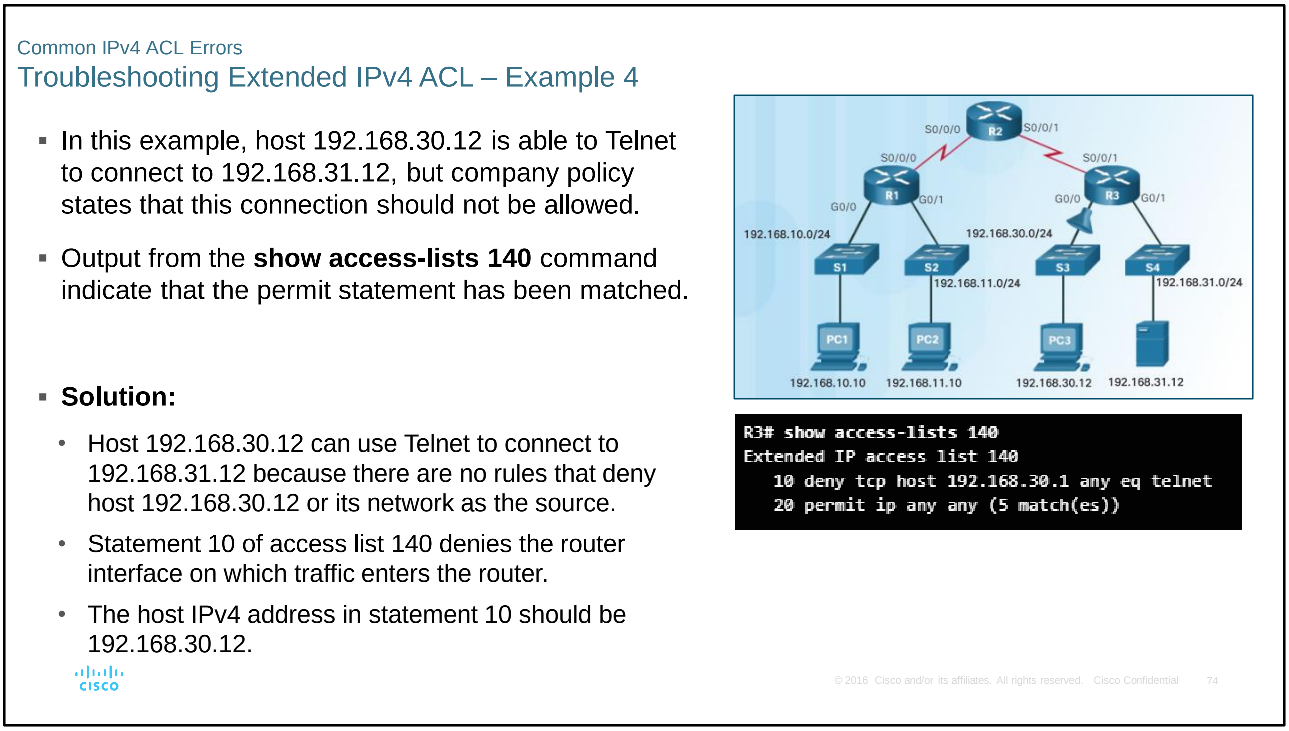
```
R1# show access-lists 130
Extended IP access list 130
  10 deny tcp any eq telnet any
  20 deny tcp 192.168.11.0 0.0.0.255 host 192.168.31.12 eq smtp
  30 permit tcp any any (12 match(es))
```

- **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**.

Troubleshooting IPv4 ACL - Example 3

In the figure, 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, **10 deny tcp 192.168.11.0 0.0.0.255 192.168.30.0 0.0.0.255 eq telnet**.



Common IPv4 ACL Errors

Troubleshooting Extended IPv4 ACL – Example 4

- 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))
```

Troubleshooting IPv4 ACL - Example 4

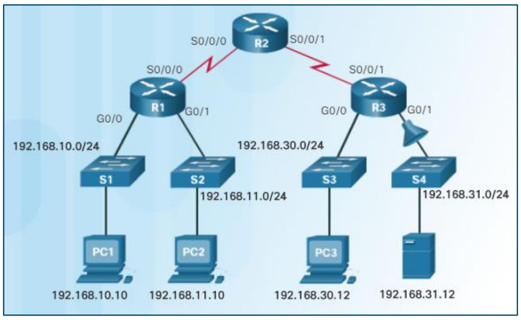
In the figure, host 192.168.30.12 is able to Telnet to connect to 192.168.31.12, but company policy states that this connection should not be allowed. Output from the **show access-lists 140** command indicate that the permit statement has been matched.

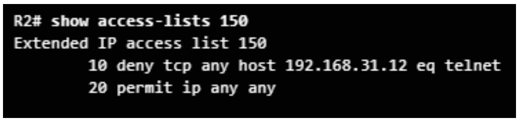
Solution - Host 192.168.30.12 can use Telnet to connect to 192.168.31.12 because there are no rules that deny host 192.168.30.12 or its network as the source. Statement 10 of access list 140 denies the router interface on which traffic enters the router. The host IPv4 address in statement 10 should be 192.168.30.12.

Common IPv4 ACL Errors

Troubleshooting Extended IPv4 ACL – Example 5

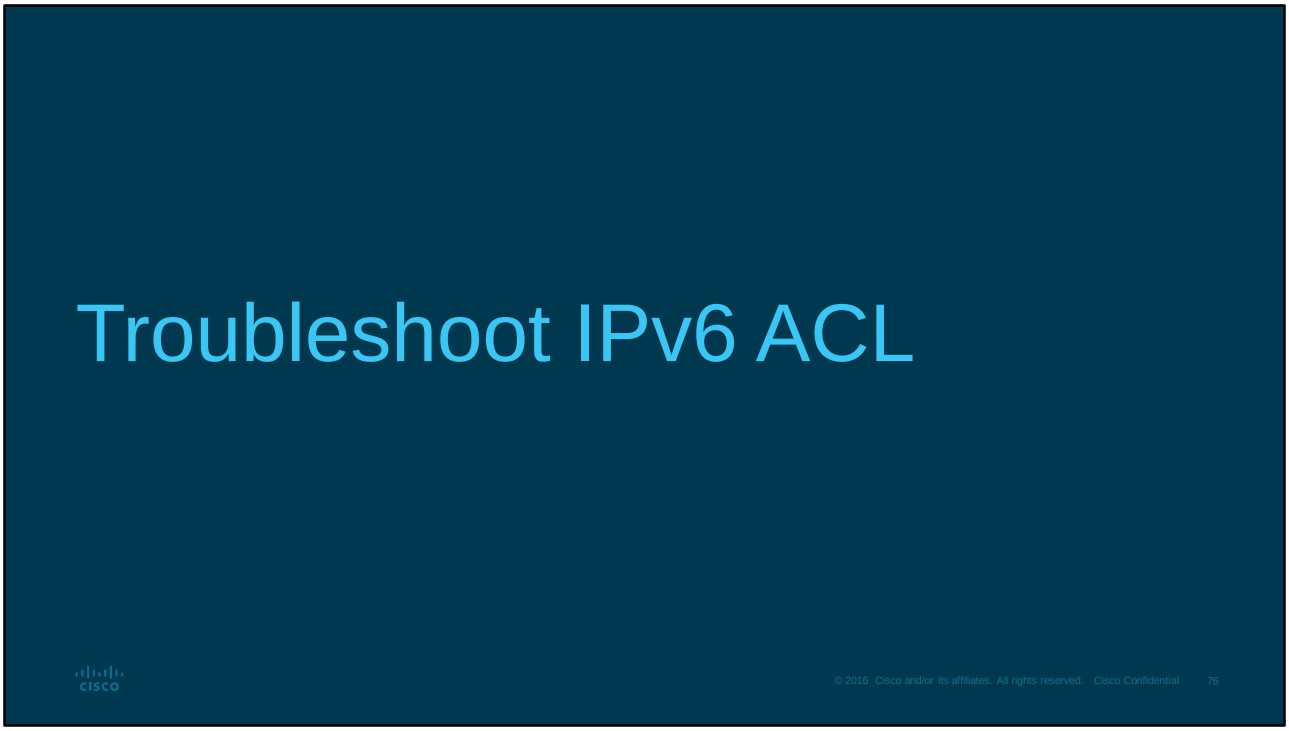
- In this example, host 192.168.30.12 can use Telnet to connect to 192.168.31.12, but according to the security policy, this connection should not be allowed.
- Output from the **show access-lists 150** command indicate that no matches have occurred for the deny statement as expected.
- **Solution:**
  - 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
```

Troubleshooting IPv4 ACL - Example 5
In the figure, 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.



Troubleshoot IPv6 ACL

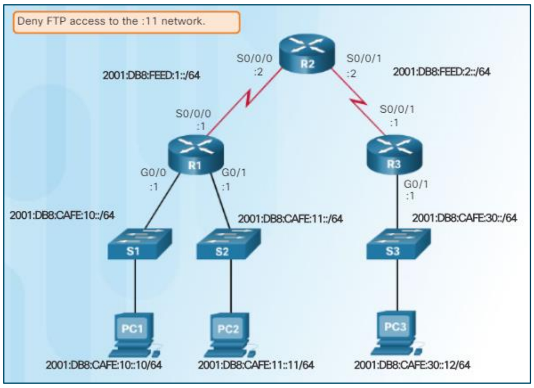
4 – Access Control Lists
4.3 – IPv6 ACL

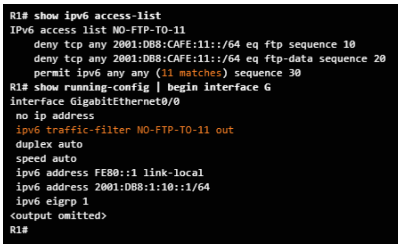
Common IPv6 ACL Errors

Troubleshooting Extended IPv6 ACL – Example 1

- 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.

- **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**.


Deny FTP access to the :11 network.


```
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#
```

Troubleshooting IPv6 ACL - Example 1

Similar to IPv4 ACL, use the **show ipv6 access-list** and **show running-config** commands to reveal typical IPv6 ACL errors.

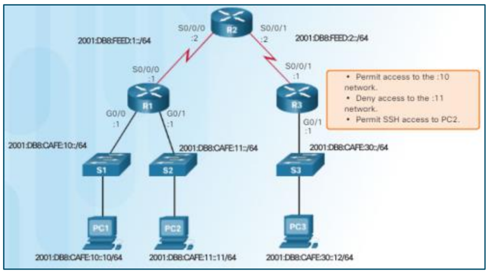
In Figure 1, R1 is configured with an IPv6 ACL to deny FTP access from the :10 network to the :11 network. However, after configuring the ACL, PC1 is still able to connect to the FTP server running on PC2. Referring to the output for the **show ipv6 access-list** command in Figure 2, matches are shown for the permit statement but not the deny statements.

Solution: The ACE in the ACL reveal no problems in their order, or in the criteria of their rules. The next step is to consider how the ACL is applied at the interface using the **ipv6 traffic-filter** command. Did the ACL get applied using the correct name, the correct interface, and in the correct direction? To check for interface configuration errors, display the running configuration, as shown in Figure 2.

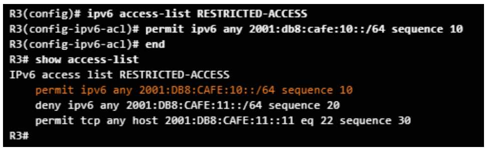
The ACL was applied using the correct name, but not the correct direction. The direction, in or out, is from the perspective of the router, meaning the ACL is currently applied to traffic before it is forwarded out the G0/0 interface and enters the :10 network. To correct the issue, remove the ipv6 traffic-filter NO-FTP-TO-11 out and replace it with ipv6 traffic-filter NO-FTP-TO-11 in. Now PC1's attempts to access the FTP server are denied, as verified with the **show ipv6 access-list** command.

Common IPv6 ACL Errors

Troubleshooting Extended IPv6 ACL – Example 2



- 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#
```

- 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.

Troubleshooting IPv6 ACL - Example 2

In the figure, R3 is configured with an IPv6 ACL named RESTRICTED-ACCESS that should enforce the following policy for the R3 LAN:

Permit access to the :10 network

Deny access to the :11 network

Permit SSH access to the PC at 2001:DB8:CAFE:11::11

However, after configuring the ACL, PC3 cannot reach the 10 network or the 11 network, and it cannot SSH into the host at 2001:DB8:CAFE:11::11.

Solution: In this situation the problem is not with how the ACL was applied. At the interface, the ACL is not misspelled, and the direction and location are correct. A close look at the IPv6 ACL reveals that the problem is with the order and criteria of the ACE rules. The first permit statement should allow access to the :10 network. However, the administrator configured a host statement and did not specify a prefix. In this case, only access to the 2001:DB8:CAFE:10:: host is allowed.

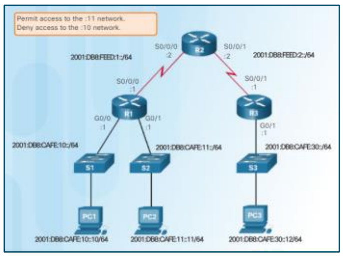
To correct this issue, remove the host argument and change the prefix to /64. You can do this without removing the ACL by replacing the ACE using the sequence number 10, as shown in Figure 2.

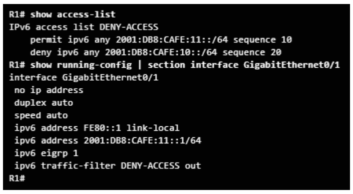
The second error in the ACL is the order of the next two statements. The policy specifies that hosts on the R3 LAN should be able to SSH into host 2001:DB8:CAFE:11::11. However, the deny statement for :11 network is listed before the permit statement. Therefore, all attempts to access the :11 network are denied before the statement permitting SSH access can be evaluated. After a match is made, no further statements are analyzed. To correct this issue, you will need to remove the statements first, and then enter them in the correct order.

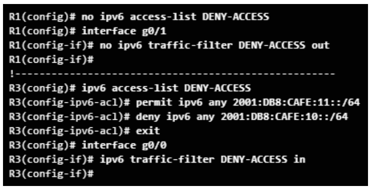
Common IPv4 ACL Errors

Troubleshooting Extended IPv6 ACL – Example 3

- 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.


Permit access to the :11 network.
Deny access to the :10 network.


```
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)#
```

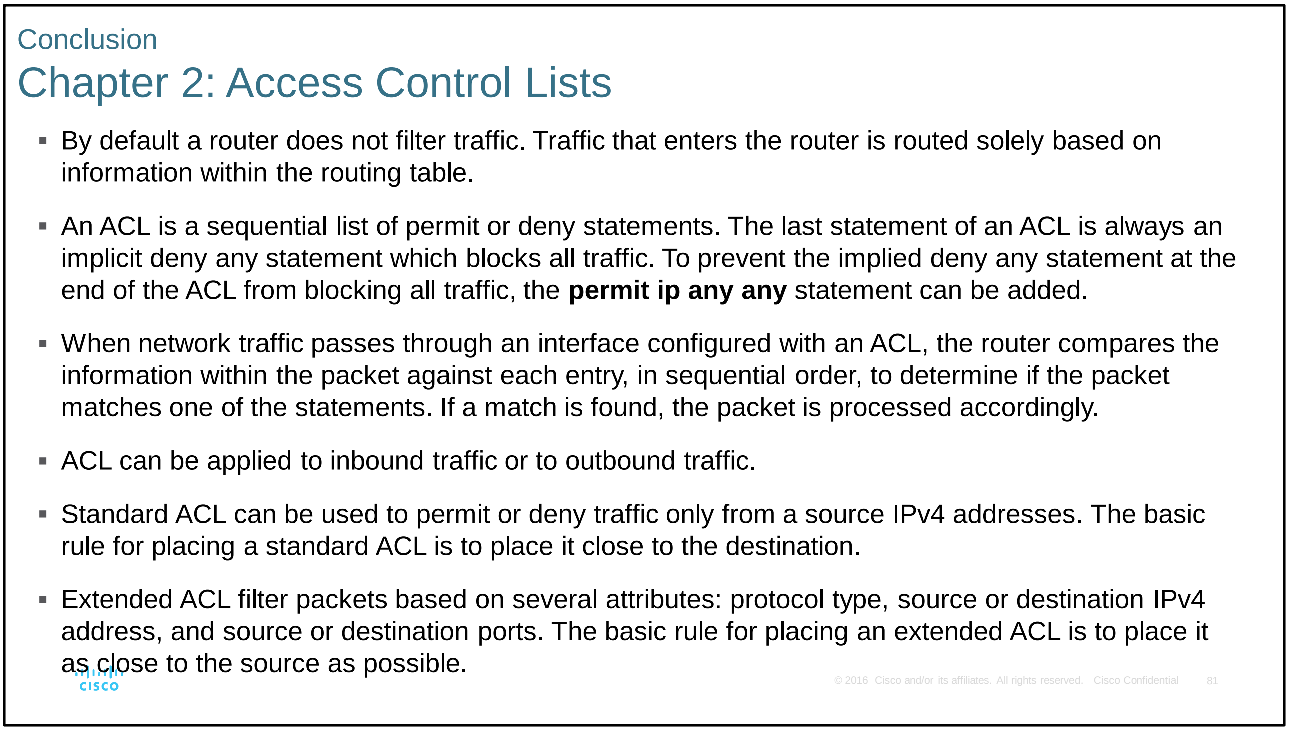
Troubleshooting IPv6 ACL - Example 3

In Figure 1, R1 is configured with an IPv6 ACL named DENY-ACCESS that should enforce the following policy for the R3 LAN:
Permit access to the :11 network from the :30 network
Deny access to the :10 network
Figure 2 shows the configuration and application of the IPv6 ACL.
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: In this situation, the problem is not with how the ACL statements were written but with the location of the ACL. Because IPv6 ACL must be configured with both a source and a destination, they should be applied closest to the source of the traffic. The DENY-ACCESS ACL was applied in the outbound direction on the R1 G0/1 interface which is closest to the destination. As a result, traffic to the :10 network is completely unaffected because it reaches the :10 network through the other LAN interface, G0/0. You could apply the ACL inbound on the R1 S0/0/0 interface. However, because we have control over R3, the best location would be to configure and apply the ACL closest to the source of the traffic. Figure 3 shows the removal of the ACL on R1 and the correct configuration and application of the ACL on R3.

Chapter Summary

80

7 – Access Control Lists
7.4 – Summary



Conclusion

Chapter 2: 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.
- ACL can be applied to inbound traffic or to outbound traffic.
- Standard ACL 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 ACL 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

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. 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.

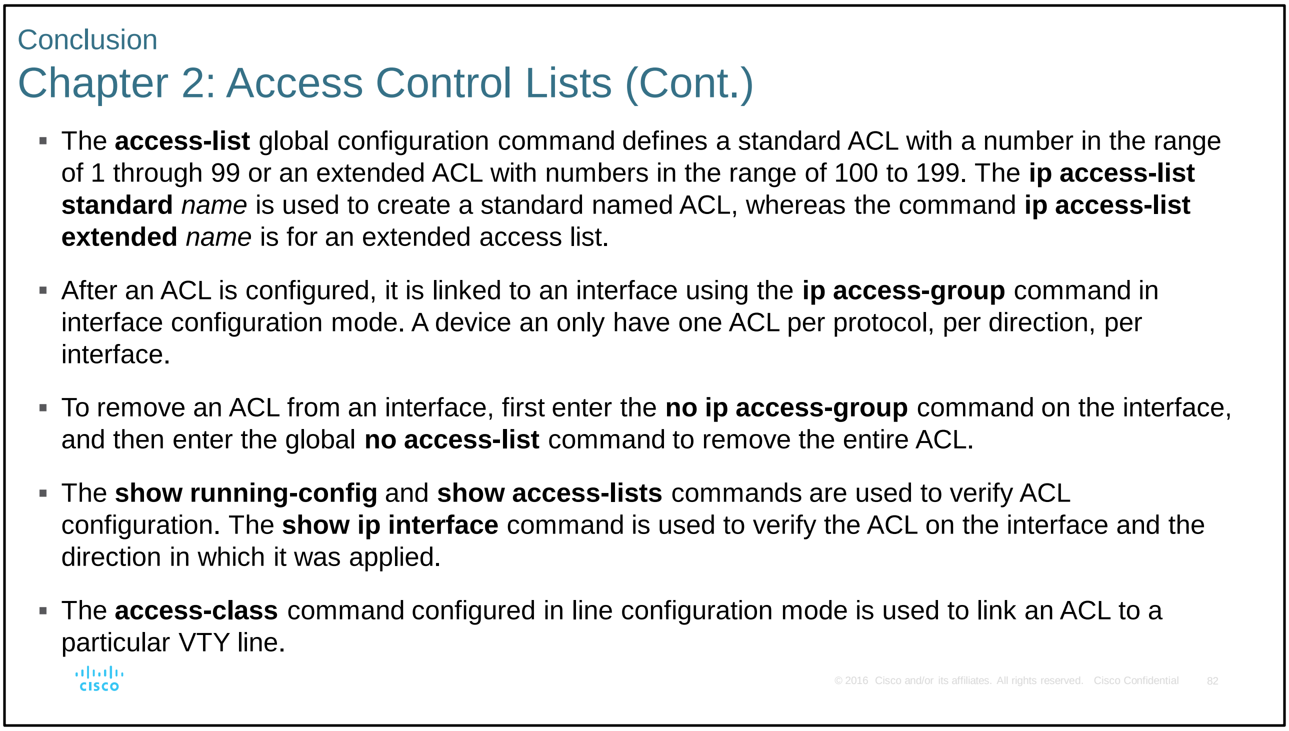
When network traffic passes through an interface configured with an ACL, the router compares the information within the packet against each entry, in sequential order, to determine if the packet matches one of the statements. If a match is found, the packet is processed accordingly.

ACL are configured to apply to inbound traffic or to apply to outbound traffic.

Standard ACL can be used to permit or deny traffic only from a 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 ACL 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.
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 and 2000 to 2699. Both standard and extended ACL can be named instead of numbered. 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 ACE 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. 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 restricts incoming and outgoing connections between a particular VTY and the addresses in an access list.
Like IPv4 named ACL, 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. Unlike IPv4 ACL, IPv6 ACL 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.

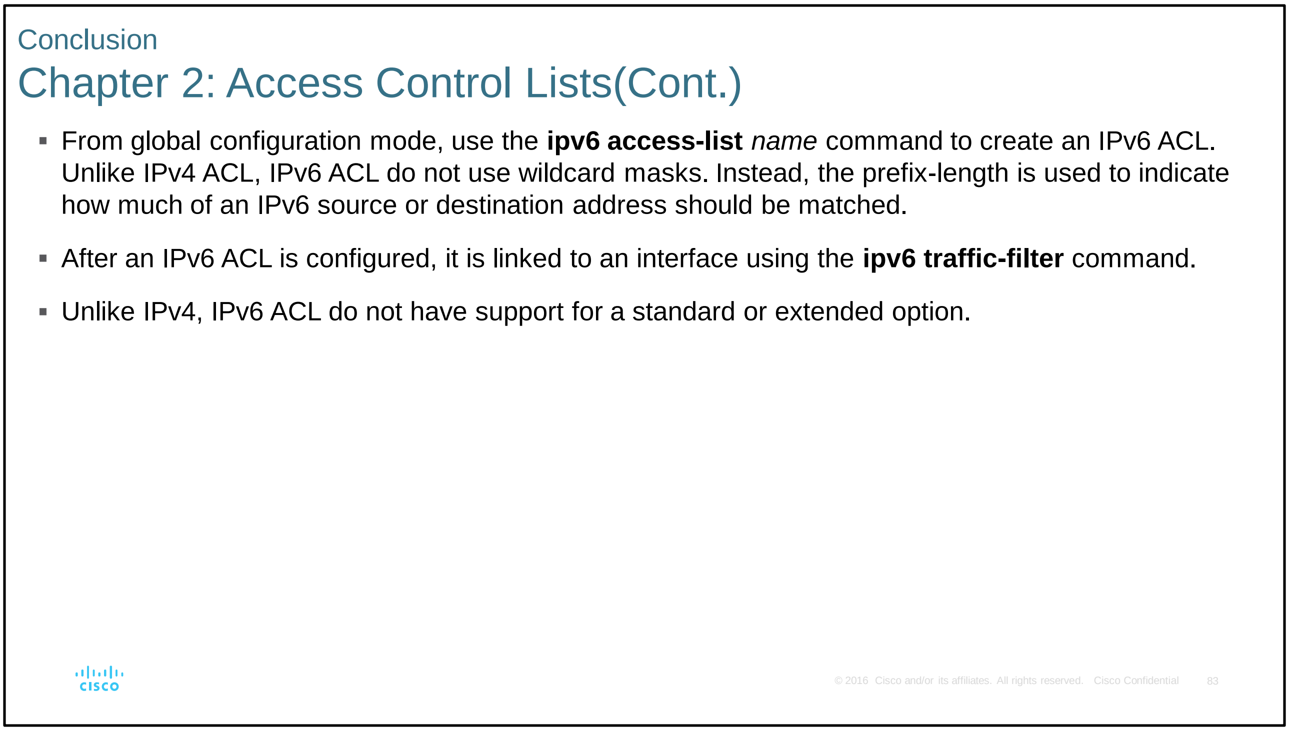


Conclusion

Chapter 2: 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.

4.5 – Summary
4.5.1 – Conclusion
4.5.1.3 – Chapter 4: Access Control Lists

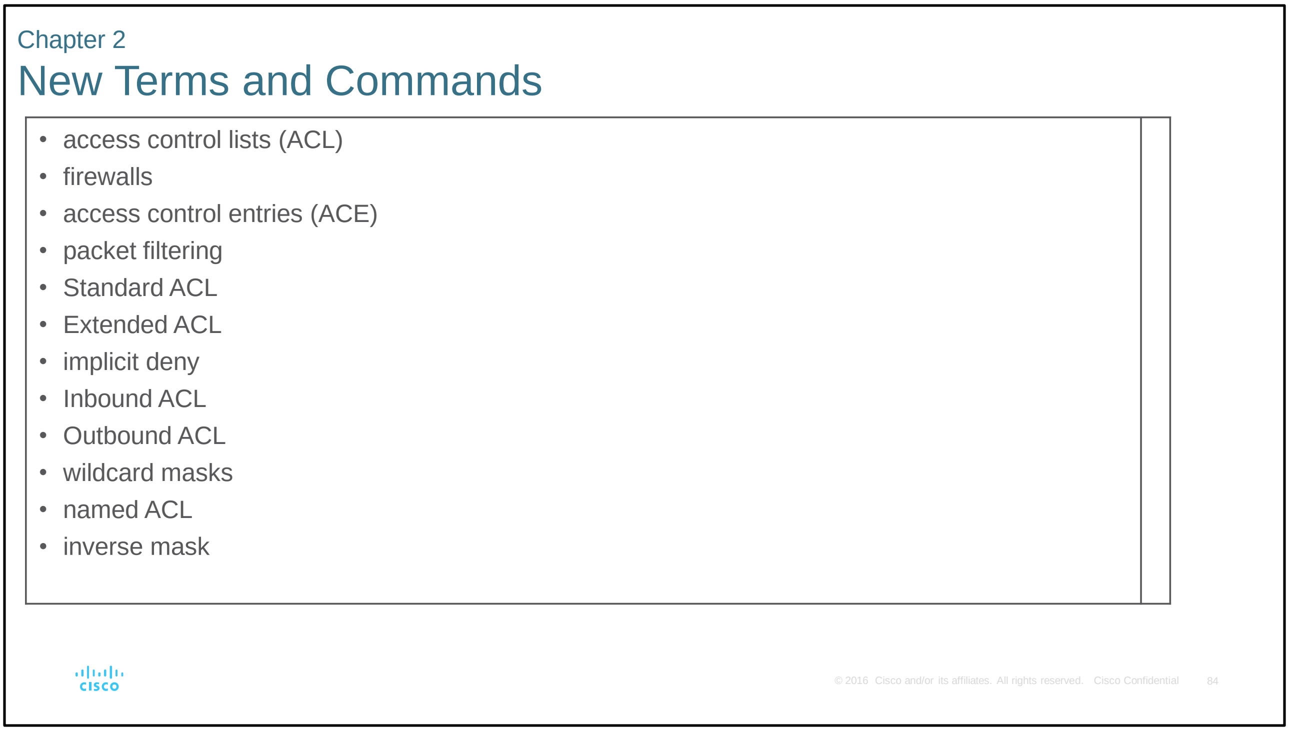


Conclusion

Chapter 2: Access Control Lists(Cont.)

- From global configuration mode, use the **ipv6 access-list** *name* command to create an IPv6 ACL. Unlike IPv4 ACL, IPv6 ACL 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 ACL do not have support for a standard or extended option.

4.5 – Summary
4.5.1 – Conclusion
4.5.1.3 – Chapter 4: Access Control Lists



Chapter 2

New Terms and Commands

- access control lists (ACL)
- firewalls
- access control entries (ACE)
- packet filtering
- Standard ACL
- Extended ACL
- implicit deny
- Inbound ACL
- Outbound ACL
- wildcard masks
- named ACL
- inverse mask

New Terms and Commands