CCNA™
CISCO® CERTIFIED NETWORK ASSOCIATE

13 IP Access Lists

CERTIFICATION OBJECTIVES

The last few chapters introduced you to routing protocols and their basic configuration. By default, once you set up routing, your router will allow any packet to flow from one interface to another. You may want to implement policies to restrict the flow of traffic, for either security or traffic policy reasons. Cisco allows you affect the flow of traffic from one interface to another by using access control lists (ACLs). ACLs, pronounced *ackles,* are a very powerful feature of the IOS. Cisco actually supports ACLs for other protocols besides IP, including IPX, XNS, DECnet, AppleTalk, and others. The remainder of this chapter focuses on IP ACLs, which are also the focus of the CCNA exam.

CERTIFICATION OBJECTIVE 13.01

ACL Overview

ACLs, known for their ability to filter traffic as it either comes into or leaves an interface, can also by used for other purposes, including the following:

- Restricting telnet (VTY) access to a router
- Filtering routing information
- Prioritizing WAN traffic with queuing
- Triggering phone calls with dial-on-demand routing (DDR), discussed in Chapter 17
- Changing the administrative distance of routes

This list contains just a small subset of ways that ACLs can be used to implement other IOS features. This chapter focuses on restricting the flow of traffic to or through a router.

Definition

ACLs are basically a set of commands, grouped together by a number or name, that are used to filter traffic entering or leaving an interface. ACL commands define specifically which traffic is permitted and which is denied. ACLs are created in *Global Configuration* mode. Once you create your group of ACL statements, you must activate them. For filtering traffic between interfaces, the ACL is activated in *Interface Subconfiguration* mode. This can be a physical interface, like `ethernet0` or `serial0`, or a logical interface, like `ethernet0.1` or `serial0.1`. When activating an ACL on an interface, you must specify in which direction the traffic should be filtered:

- Inbound (as the traffic comes into an interface)
- Outbound (before the traffic exits an interface)

exam watch

For inbound ACLs, the ACL is processed before any further processing; with outbound ACLs, the packet is routed to the interface and then the outbound ACL is processed.

With inbound ACLs, the router compares the packet to the interface ACL before the router will forward it to another interface. With outbound ACLs, the packet is received on an interface and forwarded to the exit interface; the router then compares the packet to the ACL. One restriction that ACLs have is that they cannot filter traffic that the router originates itself. For example, if you execute a ping or traceroute from the router, or if you telnet from the router to another device, ACLs applied to the router's interfaces cannot filter these connections. However, if an external device tries to ping, traceroute, or telnet *to* the router or *through* the router to a remote destination, the router can filter these packets.

Types

ACLs come in two varieties:

- Numbered and named
- Standard and extended

exam watch

Remember the filtering abilities of standard and extended ACLs as described in Table 13-1.

Numbered and named ACLs define how the router will reference the ACL. You can view this as something similar to an index value. A numbered ACL is assigned a unique number among all ACLs, whereas a named ACL is assigned a unique name among all named ACLs. These are then used by the router to filter traffic.

Each of these references to ACLs supports two types of filtering: standard and extended. Standard IP ACLs can filter only on the source IP address inside a packet, whereas an extended IP ACLs can filter on the source and destination IP addresses in the packet, the IP protocol (TCP, UDP, ICMP, and so on), and protocol information (such as the TCP or UDP source and destination port numbers).With an extended ACL, you can be very precise in your filtering. For example, you can filter a specific

TABLE 13-1

Comparing Standard and Extended ACLs

Filtered Information	Standard IP ACL	Extended IP ACL
Source address	Yes	Yes
Destination address	No	Yes
IP protocol (i.e., TCP or UDP)	No	Yes
Protocol information (i.e., port number)	No	Yes

telnet session from one of your user's PCs to a remote telnet server. Standard ACLs do not support this form of granularity. With a standard ACL, you can either permit or deny all traffic from a specific source device. Table 13-1 compares the two types of filtering for IP traffic.

Processing

ACLs are basically statements that are grouped together by either a name or a number. Within this group of statements, when a packet is processed by an ACL on the router, the router will go through certain steps in finding a match against the ACL statements.

ACLs are processed top-down by the router. Using a top-down approach, a packet is compared to the first statement in the ACL, and if the router finds a match between the packet and the statement, the router will execute one of two actions included with the statement:

- Permit
- Deny

If the router doesn't find a match of packet contents to the first ACL statement, the router will proceed to the next statement in the list, again going through the same matching process. If the second statement matches, the router executes one of the two actions. If there isn't a match on this statement, the router will keep on going through the list until it finds a match. If the router goes through the entire list and doesn't find a match, the router will *drop* the packet.

The top-down processing of ACLs brings out the following very important points:

- Once a match is found, no further statements are processed in the list.
- The order of statements is important.
- If no match is found in the list, the packet is dropped.

If there is a match on a statement, no further statements are processed. Therefore, the order of the statements is *very* important in an ACL. If you have two statements, one denying a host and one permitting the same host, whichever one appears *first* in the list will be executed and the second one will be ignored. Because order of statements is important, you should always place the most specific ACL statements at the top of the list and the least specific at the bottom of the list.

Let's take a look at an example to illustrate this process. In this example, you have an ACL on your router with two statements in this order:

1. Permit traffic from subnet 172.16.0.0/16.
2. Deny traffic from host 172.16.1.1.

Remember that the router processes these statements *top-down*. Let's assume that a packet is received on the router with a source IP address of 172.16.1.1. Given the preceding ACL, the router compares the packet contents with the first statement. Does the packet have a source address from network 172.16.0.0/16? Yes. Therefore, the result indicates that the router should permit the packet. Notice that the second statement is never processed once the router finds a match on a statement. In this example, any traffic from the 172.16.0.0/16 subnet is permitted, even traffic from 172.16.1.1.

Let's reverse the order of the two statements and see how this reordered ACL will affect traffic flow:

1. Deny traffic from host 172.16.1.1.
2. Permit traffic from subnet 172.16.0.0/16.

If 172.16.1.1 sends traffic through the router, the router first compares these packets with the first ACL statement. Since the source address matches 172.16.1.1, the router drops the packet and stops processing statements in the ACL. In this example, it doesn't matter what traffic 172.16.1.1 is sending. If another device, say 172.16.1.2, sends traffic through the router, the router compares the packet contents to the first ACL statement. Since the source address in the packet doesn't match the source address in the ACL statement, the router proceeds to the next statement in the list. Comparing the packet contents to the statement, there is a match. Therefore, the router will execute the results, permitting the traffic from 172.16.1.2.

As you can see from both of these ACL examples, the order of statements in the ACL is very important and *definitely* impacts what traffic is permitted or denied.

Implicit Deny

Another important aspect of the top-down process is that if the router compares a packet to every statement in the list and does not find a match against the packet contents, the router will *drop* the packet. This process is referred to as *implicit deny*. At the end of every ACL is an invisible statement that drops all traffic that doesn't match any of the preceding statements in the ACL. Given this process, it makes no sense to have a list of only deny statements, since the implicit deny drops all traffic anyway. Therefore, every ACL should have at least *one permit* statement; otherwise, an ACL with only deny statements will drop all traffic, given the deny statements and the hidden implicit deny statement.

exam watch

There are two actions an ACL can take: permit or deny. Statements are processed top-down. Once a match is found, no further statements are processed—therefore, order is important. If no match is found, the imaginary implicit deny statement at the end of the ACL drops the packet. An ACL should have at least one permit statement; otherwise, all traffic will be dropped because of the hidden implicit deny statement at the end of every ACL.

Important Configuration Guidelines

Configuring a access list is not a simple process. To get the configuration process right, you should be guided by the following list:

- Order of statements is important: put the most restrictive statements at the top of the list and the least restrictive at the bottom.
- ACL statements are processed top-down until a match is found, and then no more statements in the list are processed.
- If no match is found in the ACL, the packet is dropped (implicit deny).
- Each ACL needs either a unique number or a unique name.
- The router cannot filter traffic that it, itself, originates.
- You can have only one IP ACL applied to an interface in each direction (inbound and outbound)—you can't have two or more inbound or outbound ACLs applied to the same interface. (Actually, you can have one ACL for each protocol, like IP and IPX, applied to an interface in each direction.)
- Applying an empty ACL to an interface permits all traffic by default: in order for an ACL to have an implicit deny statement, you need at least one actual permit or deny statement.

exam watch

Remember the configuration guidelines.

As you can see from this list, ACLs are not a simple matter. ACLs are one of the IOS's more complex, yet powerful, features. The configuration, management, and troubleshooting of ACLs can become very complex and create many headaches for you. Therefore, it is important for you to understand the process the router uses when it compares packets to ACLs and how to create and maintain them. The following sections cover the basic configuration of ACLs on your router.

CERTIFICATION OBJECTIVE 13.02

Basic ACL Configuration

This section provides a brief introduction to the two basic commands you'll use to configure IP ACLs. The sections following this cover the actual details of configuring numbered versus named and standard versus extended ACLs.

To create an ACL, use the following command:

```
Router(config)# access-list ACL_# permit|deny conditions
```

Prior to IOS 11.2, you could give an ACL only a number as an identifier. Starting with IOS 11.2, an ACL can be referenced by a number or name. The purpose of the *ACL_#* is to group your statements together into a single list. You cannot choose just any number for an ACL. Each layer-3 protocol is assigned its own range or ranges of numbers.

exam watch

Remember the numbers you can use for IP ACLs. Standard ACLs can use numbers ranging 1–99 and 1300–1999, and extended ACLs can use 100–199 and 2000–2699.

Table 13-2 shows the valid numbers and the protocols that can use them. As you can see from this table, one advantage that named ACLs have over numbered ACLs is that with numbered ACLs, you have a limited number of lists that you can create, which is based on the range of numbers assigned to a protocol type. However, named ACLs do not have this restriction. Basically, the number of named ACLs on a router is restricted only by the amount of RAM and NVRAM your router has.

The *condition* in an ACL statement tells the router what contents in the packet need to match in order for the router to execute the action (**permit** or **deny**). The

TABLE 13-2

ACL Types and Numbers

ACL Type	ACL Numbers
IP Standard	1–99, 1300–1999
Standard Vines	1–99
IP Extended	100–199, 2000–2699
Extended Vines	100–199
Bridging type code (layer-2)	200–299
DECnet	300–399
Standard XNS	400–499
Extended XNS	500–599
AppleTalk	600–699
Bridging MAC address and vendor code	700–799
IPX Standard	800–899
IPX Extended	900–999
IPX SAP filters	1000–1099
Extended transparent bridging	1100–1199
IPX NLSP	1200–1299

condition can include matching of IP addresses and protocol information. When the router compares a packet to the condition, if it finds a match, no more ACL statements are processed; otherwise, the router proceeds to compare the packet to the next ACL statement in the list. Remember that at the end of every ACL, unseen, is the implicit deny statement.

Activating an *ACL*

Once you have built your IP ACL, it will do nothing until you apply it to a process in the IOS. This chapter focuses on filtering traffic through interfaces. Therefore, to have your router filter traffic between interfaces, you must enter the appropriate interface or interfaces and activate your ACL. Here's the command to activate it on an interface:

```
Router(config)# interface type [module_#]port_#
Router(config-if)# ip access-group ACL_# in|out
```

At the end of the **ip** `access-group` command, you must specify which ACL you are activating and in which direction:

- **`in`** As traffic comes into the interface
- **`out`** As traffic leaves the interface

In IOS 12.0 and later, you have to specify one of the two directions. In 11.3 and earlier, you did not have to enter the direction. If you omitted the direction, it defaulted to **`out`**.

Note that you can have the same ACL applied to multiple interfaces on a router, or the same ACL activated twice on the same interface: inbound and outbound. You can also apply a nonexistent ACL to an interface. This is an ACL that has no statements in it--an empty ACL will permit *all* traffic. For an ACL to have an implicit deny, it needs at least one **`permit`** or **`deny`** statement. It is highly recommended that you do *not* apply nonexistent ACLs to a router's interface. In this situation, when you create the very first statement in the list, the implicit deny is automatically placed at the bottom, which might create reachability issues for you.

exam watch

Use the `ip access-group` command to activate an ACL on an interface. You must specify the ACL number or name and the direction: either `in` or `out`.

Let's take a look at an example that has a nonexistent ACL and examine the kinds of problems that you might experience. Let's assume that you have applied an ACL (#10) to a router's `ethernet0` interface and this ACL currently doesn't have any **`permit`** or **`deny`** statements (it's empty). You are currently telnetted into the router via this interface, and your PC has an IP address of 192.168.1.1. You create an entry in ACL #10 that permits traffic from 172.16.0.0/16. As soon as you do this, you will lose your telnet connection. If you guessed that the implicit deny caused the router to drop your connection, you guessed correctly. As soon as the router has one statement in it, the implicit deny is added at the bottom. In our example, since your PC had a source address of 192.168.1.1, and this wasn't included in the first statement, the router dropped your connection because it couldn't find any matching statements in ACL #10.

Editing Entries

As you can see in the last section, creating and maintaining an ACL can be a complex process. This section covers some of the editing basics that you should know when adding, modifying, or deleting ACL statements.

First, you cannot delete a specific entry in an ACL—you can only delete the entire list. This statement is true with numbered ACLs, but not true with named ACL statements, as you will see later on in this chapter. To delete an ACL, use the **no** access-list command, followed by the number of the ACL. This deletes the entire list. If you try to delete a specific entry in the list, the router processes only the first three parameters of the command: **no** access-list *ACL_#*. Second, you cannot insert an entry at the beginning or middle of an access list. Whenever you enter an ACL command on the command line, the command is always added at the *end* of the list. And third, you cannot modify an existing entry in an ACL.

You will, at some point in time, need to either add, delete, or modify an entry in an ACL. Given the preceding issues, you will need to perform the following steps in order to easily manage the editing process of your list:

1. Execute the **show** running-config command and scroll down to your router's ACL entries.
2. Use your mouse to select and copy the ACL commands.
3. Past the copied ACL commands into a text editor, such as Notepad.
4. Edit your ACL in the text editor, adding entries, deleting entries, and modifying entries.
5. Select and copy the ACL in your text editor.
6. On the router, remove the application of the ACL on the interface: **no** ip access-group *ACL_#* **in|out**.
7. Delete the old access list: **no** access-list *ACL_#*.
8. Past the ACL from your text editor into *Configuration* mode. When you do this, the router accepts and processes each statement individually. If there is a syntax problem with an ACL command, the router will tell you. If this is the case, go back to step 4.
9. Reactivate the ACL on your router's interface with the **ip** access-group *Interface Subconfiguration* mode command.

exam watch ***Be familiar with the steps to edit an ACL on a router.***

I've used this procedure successfully for many years. If you attempt to fix ACL problems from the CLI, you are just opening yourself up to a lot of headaches. For instance, if you delete your ACL and reenter it manually, and you make a mistake on the very last command, you'll need to delete the whole ACL and start over again.

CERTIFICATION OBJECTIVE 13.03

Wildcard Masks

When dealing with IP addresses in ACL statements, you can use wildcard masks to match on a range of addresses instead of having to manually enter every IP address that you want to match on. Wildcard masks were briefly discussed under the heading "OSPF" in Chapter 11. This section goes into more depth about wildcard masks and how they are used in ACLs.

First, a wildcard mask is *not* a subnet mask. Like an IP address or a subnet mask, a wildcard mask is composed of 32 bits. Table 13-3 compares the bit values in a subnet mask and a wildcard mask. With a wildcard mask, a 0 in a bit position means that the corresponding bit position in the address of the ACL statement *must* match the bit position in the IP address in the examined packet. A 1 in a bit position means that the corresponding bit position in the address of the ACL statement does *not* have to match the bit position in the IP address in the examined packet. In other words, the wildcard mask and the address in the ACL statement work in tandem. The wildcard mask tells the router which addressing bits must match in the address of the ACL statement.

In reality, a wildcard mask is more like an *inverted* subnet mask. For instance, if you want to match on any address in a subnet or network, all you need to do is to take the subnet mask, invert its bit values (change the 1's to 0's and the 0's to 1's), and you have a corresponding wildcard mask. Let's look at a simple example of performing a binary conversion of a subnet mask to a wildcard mask. Let's assume that you have subnet mask of 255.255.0.0. Its binary representation is 11111111.11111111.0000000.00000000. When you convert this to a wildcard mask, invert the bits, like this: 00000000.00000000.11111111.11111111.

Then covert this to decimal: 0.0.255.255. This is the corresponding wildcard mask for the subnet mask of 255.255.0.0. In this example, the wildcard mask tells the router that the first 16 bits of the corresponding IP address in the ACL statement must match the contents in the IP address of the packet for the router to continue processing the statement; otherwise, the router will proceed to the next ACL statement. As you can see, this was an example that was easy to convert.

TABLE 13-3 Subnet Mask Versus Wildcard Mask

Bit Value	Subnet Mask	Wildcard Mask
0	Host component	Must match
1	Network component	Ignore

Let's look at a more difficult example. Let's assume that you want to match on a subnet that has a subnet mask of 255.255.240.0. Here's the entire subnet mask in binary: 11111111.11111111.11110000.00000000.

In this example, the first, second, and fourth octets are easy to convert: the difficult conversion is in the third octet. To convert the subnet mask to a wildcard mask, invert all of the bits, as is shown here: 00000000.00000000.00001111.11111111.

Next convert this back to decimal. This results in a wildcard mask of 0.0.15.255. As you can see from the last two examples, if a subnet mask has 0 in an octet, the wildcard mask has a value of 255; and if the subnet mask has 255 in an octet, the wildcard mask has a value of 0. However, the third octet in the last example makes this process more difficult.

In reality, I've developed a shortcut to alleviate the conversion of a subnet mask to a wildcard mask. When doing the conversion, subtract each byte in the subnet mask *from* 255. The result will be the corresponding byte value for the wildcard mask. Going back to the 255.255.240 example, here is the short cut:

- First byte: 255 – 255 (first subnet byte value) = 0 (wildcard mask value)
- Second byte: 255 – 255 (second subnet byte value) = 0 (wildcard mask value)
- Third byte: 255 – 240 (third subnet byte value) = 15 (wildcard mask value)
- Fourth byte: 255 – 0 (fourth subnet byte value) = 255 (wildcard mask value)

As you can see, this results in a wildcard mask of 0.0.15.240. This simple trick makes converting subnet masks to wildcard masks very easy.

exam watch

Wildcard masks are used to match against bits in a packet. A 0 in a bit position means match, and a 1 means ignore. If you want to match against a subnet, take the corresponding subnet mask and invert it. The trick is to subtract each octet in the mask from 255, resulting in the wildcard mask.

Special Wildcard Masks

There are two special types of wildcard masks:

- 0.0.0.0
- 255.255.255.255

A wildcard mask of 0.0.0.0 tells the router that all 32 bits of the address in the ACL statement must match those found in the IP packet in order for the router to execute the action for the statement. A 0.0.0.0 wildcard mask is called a *host mask*. Here's a simple example of this information in an ACL statement: 192.168.1.1 0.0.0.0. This statement tells the router to look for the exact same IP address (192.168.1.1) in the IP packet. If the router doesn't find a match, the router will go to the next ACL statement. If you configure 192.168.1.1 0.0.0.0 on your router, the router will covert this to the following: **`host`** `172.16.1.1`. Note the keyword **`host`** that precedes the IP address.

exam watch

Be familiar with how wildcard masks work, as well as the special notation Cisco uses for a match on all devices or a specific host, as shown in Table 13-4.

A wildcard mask of 255.255.255.255 tells the router the exact opposite of a 0.0.0.0 mask. In this mask, all of the bit values are 1's, which tells the router that it doesn't matter what is in the packet that it is comparing to the ACL statement—*any* address will match. Typically, you would record this as an IP address of 0.0.0.0 and a wildcard mask of 255.255.255.255, like this: 0.0.0.0 255.255.255.255. If you enter this, the router will cover the address and mask to the keyword **`any`**. Actually, the IP address that you enter with this mask doesn't matter. For instance, if you enter 192.168.1.1 255.255.255.255, this still matches any IP address. Remember that it's the wildcard mask that determines what bits in the IP address are *interesting* and should match.

Examples

Since the concept of a wildcard mask can be confusing, let's look at some examples. Table 3-4 shows some examples of addresses and wildcard masks.

TABLE 13-4 Wildcard Mask Examples

IP Address	Wildcard Mask	Matches
0.0.0.0	255.255.255.255	Match on any address (keyword **`any`**).
172.16.1.1	0.0.0.0	Match only if the address is 172.16.1.1 (preceded by the keyword **`host`**).
172.16.1.0	0.0.0.255	Match only on packets that are in 172.16.1.0/24 (172.16.1.0–172.16.1.255)
172.16.2.0	0.0.1.255	Match only on packets that are in 172.16.2.0/23 (172.16.2.0–172.16.3.255)
172.16.0.0	0.0.255.255	Match only on packets that are in 172.16.0.0/16 (172.16.0.0–172.16.255.255)

CERTIFICATION OBJECTIVE 13.04

Types of ACLs

The following sections cover the configuration of both numbered and named ACLs. The first two sections deal with configuring numbered standard and extended ACLs; they are followed by a section on configuring named ACLs and then a section on how to verify your ACL configuration.

Standard Numbered ACLs

Standard IP ACLs are simple and easy to configure. First, standard IP ACLs filter on only the *source IP address* in an IP packet. Use the following command to create an entry in a standard numbered IP ACL:

```
Router(config)# access-list 1-99|1600-1999 permit|deny
                    source_IP_address
                      [wildcard_mask] [log]
```

exam watch ***Be very familiar with the syntax of a standard ACL, as well as the fact that it can filter only on source addresses in a packet.***

exam watch ***If you omit the wildcard mask in a standard ACL, it defaults to 0.0.0.0 (an exact match is required).***

With a standard numbered IP ACL, you can use list numbers of 1–99 and 1600–1999. Following this is the action the router should take if there is a match on the condition. The condition is based solely on the source IP address. You enter this followed by an optional wildcard mask. If you omit the mask, it defaults to 0.0.0.0—an exact match is required in order to execute the action.

Following this is the optional **log** parameter, which is new to standard ACLs in IOS 12.0. This parameter will cause any match of this statement to be printed to the console port of the router. These messages, by default, will not appear on a telnet connection to the router unless you execute the following:

```
Router# terminal monitor
```

exam watch

Use the `terminal monitor` command to view console output on nonconsole connections.

You can also forward these messages to a syslog server. This setup is useful for debugging and security purposes.

Activating a Standard IP ACL

Once you have created your ACL, you can proceed to activate it on a router's interface with the following configuration:

```
Router(config)# interface type [module_#]port_#
Router(config-if)# ip access-group ACL_# in|out
```

In IOS version 12.0 and later, you *must* specify either **`in`** or **`out`**. In previous versions, you could omit this and it would default to **`out`**.

Standard IP ACL Examples

Now that you have been introduced to the two basic commands to create and activate a standard numbered IP ACL, let's look at some examples to help you further your understanding. Here's the first example:

```
Router(config)# access-list 1 permit 192.168.1.1
Router(config)# access-list 1 deny 192.168.1.2
Router(config)# access-list 1 permit 192.168.1.0 0.0.0.255
Router(config)# access-list 1 deny any
Router(config)# interface serial 0
Router(config-if)# ip access-group 1 in
```

In this example, the first ACL statement in ACL #1 says that in order to execute the **`permit`** action, the IP packet must have a source address of 192.168.1.1—if it doesn't, the router proceeds to the second statement. Remember that if you omit the wildcard mask on a standard ACL, it defaults to 0.0.0.0—an exact match of the corresponding address in the ACL statement. The second ACL statement says that in order to execute the **`deny`** action, the IP packet must have a source address of 192.168.1.2; if it doesn't, the router proceeds to the third statement. The third ACL statement says that in order to execute the **`permit`** action, the IP packet must have a source address between 192.168.1.0 and 192.168.1.255—if it doesn't, the router proceeds to the fourth statement. The fourth statement is actually not necessary: it drops any packet. You don't need this statement, since there is an invisible implicit deny any statement at the end of every ACL. The last two commands in the ACL example activate ACL #1 on `serial0` as traffic comes into the interface.

Actually, you could have written the preceding ACL like this:

```
Router(config)# access-list 1 deny 192.168.1.2
Router(config)# access-list 1 permit 192.168.1.0 0.0.0.255
Router(config)# interface serial 0
Router(config-if)# ip access-group 1 in
```

This example reduces your configuration from four ACL statements in the list down to two, which increases the performance of your router.

Here's another example of a standard ACL:

```
Router(config)# access-list 2 deny 192.168.1.0
Router(config)# access-list 2 deny 172.16.0.0
Router(config)# access-list 2 permit 192.168.1.1
Router(config)# access-list 2 permit 0.0.0.0 255.255.255.255
Router(config)# interface ethernet 0
Router(config-if)# ip access-group 1 out
```

This ACL example has a few problems with it. Examine it and see if you can spot them.

The first ACL statement appears to deny all traffic from 192.168.1.0/24. In reality, it will accomplish nothing. Remember that if you omit the wildcard mask for the address, it defaults to 0.0.0.0—an exact match. The problem with this is that you'll never have a packet with a source address of 192.168.1.0, since this is a network number, and not a host address. The second statement has the same problem. The third and fourth statements are okay.

As you can see, configuring ACLs can be tricky. For the preceding example, here's the updated configuration:

```
Router(config)# access-list 2 deny 192.168.1.0 0.0.0.255
Router(config)# access-list 2 deny 172.16.0.0 0.0.255.255
Router(config)# access-list 2 permit 192.168.1.1
Router(config)# access-list 2 permit 0.0.0.0 255.255.255.255
Router(config)# interface ethernet 0
Router(config-if)# ip access-group 1 out
```

In this example, the first statement now says that any packet with a source address from network 192.168.1.0/24 should be dropped. The second statement will drop any traffic from the class B network 172.16.0.0/16. The third statement will permit traffic from 192.168.1.1. The fourth statement will permit traffic from anywhere. Actually, there is *still* a problem with this configuration—look at the first and third statements. Will the third statement ever be executed? If you answered *no*, then you would be correct. In this situation, you need to put the more specific entry before the less specific. Another minor point to make is that the fourth statement in the list could represent the address as the keyword **any**. Here's the updated configuration:

```
Router(config)# access-list 2 permit 192.168.1.1
Router(config)# access-list 2 deny 192.168.1.0 0.0.0.255
Router(config)# access-list 2 deny 172.16.0.0 0.0.255.255
Router(config)# access-list 2 permit any
Router(config)# interface ethernet 0
Router(config-if)# ip access-group 1 out
```

exam watch

Be familiar with tricky ACL configurations like the preceding example.

There's actually one more problem with this ACL. If you guessed the ACL number used on the interface is not correct, then you guessed correctly. Notice that the ACL created has a number of 2, while the application of the ACL on the interface uses 1. To fix this, use the following configuration:

```
Router(config)# interface ethernet 0
Router(config-if)# no ip access-group 1 out
Router(config-if)# ip access-group 2 out
```

Note that you must first remove the old ACL from the interface before applying the new ACL.

CertCam

13.01. The CD contains a multimedia demonstration of configuring a standard numbered ACL on a router.

Restricting Telnet Access to the Router

Besides using standard IP ACLs to filter traffic as it enters and/or leaves an interface, you can also use them to restrict telnet access *to* your router. You might want to do this to allow only network administrators to telnet into your router. Setting this up is almost the same as what you would do to restrict access on an interface.

First, you need to create a standard ACL that has a list of **permit** statements that allow your corresponding network administrators telnet access; include the IP addresses of their PCs in this list. Next, you need to activate your ACL. However, you will not do this on any of the router's interfaces. If you were to activate this ACL on an interface, it would allow any type of traffic from your administrators but drop *all* other traffic. As you may recall from Chapter 5, when someone telnets into your router, the router associates this connection with a virtual terminal (VTY) line. Therefore, you'll apply your standard ACL to the VTYs, like this:

```
Router(config)# line vty 0 4
Router(config-line)# access-class standard_ACL_# in|out
```

Remember that your router supports five telnets by default (0–4). You can configure all VTYs simultaneously by specifying the beginning and ending line numbers after the **vty** parameter. If you don't apply the restriction to all of your VTYs, then you are leaving a backdoor into your router, which might cause a security problem.

Also, notice the command used to apply the ACL to the line: **access-class**. This is different from activating an ACL on a router's interface. If you use the **in** parameter, you are restricting telnet access to the router itself. The **out** parameter is kind of unique. By using this parameter, you are restricting what destinations this router can telnet *to* when someone uses the **telnet** or **connect** commands. This creates an exception to a standard ACL and has the router treat the address in the ACL statements as a destination address; it causes the router to compare this address to the address in the **telnet** command before allowing the user on the router to telnet to the specified destination.

Here's a simple example of using a standard ACL to filter telnet traffic to a router:

```
Router(config)# access-list 99 permit 192.168.1.0 0.0.0.255
Router(config)# line vty 0 4
Router(config-line)# access-class 99 in
```

In this example, only traffic from 192.168.1.0/24 is allowed to telnet in this router. Because of the implicit deny at the end of **access-list** 99, all other telnets to this router will be dropped.

As you will see in the next section, you can also use extended ACLs to restrict access to the router; but this configuration is much more complex. Second, extended ACLs are applied to interfaces and thus won't be able to restrict telnet access *from* the router to a remote destination. And third, whenever you apply an ACL to an interface on the router, you'll affect the performance of the router on that interface. Depending on the router model, the IOS version, and the features you have enabled, the degradation in performance will vary. Therefore, if you only want to restrict telnet access to or from the router, using a standard ACL and the **access-class** statement on your VTYs is the best approach.

CertCam

13.02. The CD contains a multimedia demonstration of configuring a standard numbered ACL to restrict telnet access on a router.

exam watch

You can restrict telnets to your router by applying a standard ACL to the VTY lines on your router. You need to apply them with the `access-class` Line Subconfiguration mode command. Please note that you can also do this with an Extended ACL, but this requires more configuration on your part.

EXERCISE 13-1

ON THE CD

Configuring Standard Numbered ACLs

These last few sections dealt with the configuration of standard numbered ACLs. This exercise will help you reinforce this material by configuring a standard numbered ACL on a router to restrict access through it. You'll perform this lab using Boson's NetSim™ simulator. This exercise has you first set static routes two routers (2600 and 2500) and verify network connectivity. Following this, you'll configure your ACL. You can find a picture of the network diagram for Boson's NetSim™ simulator in the Introduction of this book. After starting up the simulator, click on the *LabNavigator* button. Next, double-click on *Exercise 13-1* and click on the *Load Lab* button. This will load the lab configuration based on Chapter 5's and 7's exercises.

1. On the 2500, configure a static route to 192.168.1.0/24, which is off of the 2600. View the routing table.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2500*. Configure the static route: **`configure`** `terminal`, **`ip`** `route 192.168.1.0 255.255.255.0 192.168.2.1`, and **`end`**. View the static route: **`show`** `ip route`. Make sure that 192.168.1.0/24 shows up in the routing table as a static route (`S`).

2. On the 2600, configure a static route to 192.168.3.0/24, which is off of the 2500. View the routing table.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. Configure the static route: **`configure`** `terminal`, **`ip`** `route 192.168.3.0 255.255.255.0 192.168.2.2`, and **`end`**. View the static route: **`show`** `ip route`. Make sure that 192.168.3.0/24 shows up in the routing table as a static route (`S`).

3. From Host3, test connectivity to the 2600 and Host1.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host3*. Ping the `serial0` and `fa0/0` interface of the 2600 router: **`ping`** `192.168.2.1` and **`ping`** `192.168.1.1`. The pings should be successful. Ping Host1: **`ping`** `192.168.1.10`. The ping should be successful.

4. Check network connectivity between the 2950-1 switch, the 2500 router, and the 2600 router.

   At the top of the simulator in the menu bar, click on the *eSwitches* icon and choose *2950-1*. From the 2950-1 switch, ping the 2600 router: **`ping`** `192.168.1.1`. At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2500*. From the 2500 router, ping the 2600 router: **`ping`** `192.168.1.1`. At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. From the 2600 router, ping the 2950-1 switch: **`ping`** `192.168.1.4`. From the 2600 router, ping the 2500 router: **`ping`** `192.168.2.2`.

5. Configure a standard numbered ACL on the 2600 to allow traffic from the 2950-1 switch to the 2600, but to deny all other traffic. Enable logging of all traffic for the ACL statements.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. On the 2600, create a standard ACL statement to permit access from the 2950-1 switch, logging matches: **`configure`** `terminal` and **`access-list`** `1 permit 192.168.1.4 0.0.0.0 log`. Create a second ACL statement to deny all traffic, logging matches: **`access-list`** `1 deny any log`. Exit configuration mode: **`end`**. Examine the ACL configuration: **`show`** `access-lists`.

6. Activate the ACL on the 2600 router on `fa0/0`.

   Activate the ACL on the 2600 router by applying the ACL to the VTY lines: **`configure`** `terminal` and **`interface`** `fa0/0`. Apply the ACL: **`ip`** `access-group 1 in`.

7. Test the ACL from the 2950-1.

   At the top of the simulator in the menu bar, click on the *eSwitches* icon and choose *2950-1*. From the 2950-1 switch, ping the 2600: **`ping`** `192.168.1.1`. The ping should be successful. Examine the ACL matches on the 2600. At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600* and then **`show`** `access-lists`. There should be five matches on the **`permit`** statement.

8. Test the ACL from the 1900-1.

   At the top of the simulator in the menu bar, click on the *eSwitches* icon and choose *1900-1*. From the 1900-1 switch, ping the 2600: **ping** `192.168.1.1`. The ping should fail. Examine the ACL matches on the 2600: At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600* and **show** `access-lists`. There should be five matches on the **deny** statement.

9. Remove the ACL configuration from the router.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. On the 2600 router, remove the application of the ACL. Go into the interface: **configure** `terminal` and **interface** `fa0/0`. Deactivate the ACL: **no** `ip access-group 1 in`. Go back to *Global Configuration* mode: **exit**. Delete the ACL statements: **no** `access-list 1`. Exit configuration mode: **end**. Use the **show** `access-list` command to verify the ACL no longer exists.

10. Test connectivity from both switches.

    At the top of the simulator in the menu bar, click on the *eSwitches* icon and choose *2950-1*. From the 2950-1 switch, ping the 2600: **ping** `192.168.1.1`. At the top of the simulator in the menu bar, click on the *eSwitches* icon and choose *1900-1*. The ping should be successful. From the 1900-1 switch, ping the 2600: **ping** `192.168.1.1`. The ping should also be successful.

Now you should be more comfortable with configuring standard numbered ACLs on a router.

Extended Numbered ACLs

Extended IP ACLs are much more flexible in what you can match on than standard ACLs. Extended ACLs can match on all of the following information:

- Source *and* destination IP addresses
- IP protocol—IP, TCP, UDP, ICMP, and so on
- Protocol information, such as port numbers for TCP and UDP, or message types for ICMP

The following sections cover the configuration and use of extended numbered IP ACLs.

Command Syntax

Here is the generic command to configure an extended numbered IP ACL:

```
Router(config)# access-list 100-199|2000-2699 permit|deny
                    IP_protocol
                    source_address source_wildcard_mask
                        [protocol_information]
                    destination_address destination_wildcard_mask
                        [protocol_information] [log]
```

As you can see from this command, the configuration of an extended ACL is more complicated than that of a standard one. Extended IP numbered ACLs can use list numbers in the ranges 100–199 and 2000–2699. After the action (**permit** or **deny**) comes the IP protocol that you want to match on. This is the first major difference between an extended ACL and a standard one. These IP protocols include the following: **ip**, **icmp**, **tcp**, **gre**, **udp**, **igrp**, **eigrp**, **igmp**, **ipinip**, **nos**, and **ospf**. If you want to match on any IP protocol—TCP, UDP, ICMP, and so on—use the **ip** keyword for the protocol. For the CCNA exam, you'll want to focus on the **ip**, **icmp**, **tcp**, and **udp** parameters.

exam watch

Be very familiar with the general syntax of an extended ACL statement.

The second major difference is that you must specify both the source *and* destination addresses *and* wildcard masks. With a standard ACL, you can specify only the source address, and the wildcard mask is optional. Depending on the IP protocol, you might be able to add additional protocol information for the source and/or destination. For example, TCP and UDP allow you to specify both source and destination port numbers, and ICMP allows you to specify ICMP message types. As with standard ACLs, you can log messages to the console or a logging server with the **log** parameter.

TCP and UDP

Use the following syntax to configure an extended ACL for TCP or UDP.

```
Router(config)# access-list 100-199|2000-2699 permit|deny
                    tcp|udp
                    source_address source_wildcard_mask
                        [operator source_port_#]
                    destination_address destination_wildcard_mask
                        [operator destination_port_#]
                    [established] [log]
```

After specifying the action (**permit** or **deny**), you configure the IP protocol: **tcp** or **udp**.

exam watch

Know the syntax of an extended ACL statement when filtering TCP or UDP traffic.

Operators With TCP and UDP, you can specify the source, destination, or both source and destination port numbers or names. To specify how to perform the match, you must configure an *operator*. The operator tells the router how to match on the port number or numbers. Table 3-5 lists the valid operators for TCP and UDP connections. Note that these operators apply only to TCP and UDP connections. Other IP protocols do not use them.

Ports Numbers *and Names* For TCP and UDP connections, you can list either the name of the port or the number of the port. For example, if you wanted to match on telnet traffic, you could use either the keyword **telnet** or the number **23**. Table 3-6 lists some of the most common port names and numbers for TCP connections.

Here is the complete list of TCP port names that you can use: **bgp**, **chargen**, **daytime**, **discard**, **domain**, **echo**, **finger**, **ftp**, **ftp-data**, **gopher**,

TABLE 13-5

TCP and UDP Operators

Operator	Explanation
lt	Less than
gt	Greater than
neq	Not equal to
eq	Equal to
range	Range of port numbers

TABLE 13-6

Common TCP Port Names and Numbers

Port Name	Command Parameter	Port Number
FTP Data	**ftp-data**	**20**
FTP Control	**ftp**	**21**
Telnet	**telnet**	**23**
SMTP	**smtp**	**25**
WWW	**www**	**80**

hostname, **irc**, **klogin**, **kshell**, **lpd**, **nntp**, **pop2**, **pop3**, **smtp**, **sunrpc**, **syslog**, **tacacs-ds**, **talk**, **telnet**, **time**, **uucp**, **whois**, and **www**. The name **pop3** is commonly used by e-mail clients to access their e-mail from an e-mail server; **www** is a web connection to an HTTP web server. If you don't find the port name in this list, you can still specify the port by its number. If you omit the port number or name, then the ACL looks for a match on all TCP connections.

Table 3-7 shows some of the common UDP port names and numbers.

exam watch

Here is the complete list of UDP port names that you can use: biff, bootpc, bootps, discard, dns, dnsix, echo, mobile-ip, nameserver, netbios-dgm, netbios-ns, ntp, rip, snmp, snmptrap, sunrpc, syslog, tacacs-ds, talk, tftp, time, who, and xdmcp. If you don't find the port name in this list, you can still specify the port by its number. If you omit the port number or name, then the ACL looks for a match on all UDP connections.

exam watch

Understand the use of the established keyword with TCP ACL statements.

established *Keyword*

The **established** keyword is used only for TCP connections. The assumption behind the use of this keyword is that you are originating TCP traffic on the inside of the network and filtering the returning traffic as it comes back into your network. In this situation, this keyword allows (or denies) any TCP traffic that has the RST or ACK bit set in the TCP segment header. Refer to Chapter 2 for an explanation of connection-oriented transport protocols and Chapter 3 for the mechanics of TCP.

CertCam

13.03. The CD contains a multimedia demonstration of configuring an extended numbered ACL to allow telnet traffic through a router.

TABLE 13-7 Common UDP Port Names and Numbers

Port Name	Command Parameter	Port Number
DNS Query	`dns`	53
TFTP	`tftp`	69
SNMP	`snmp`	161
IP RIP	`rip`	520

ICMP

The following command shows the syntax of filtering ICMP traffic:

```
Router(config)# access-list 100-199|2000-2699 permit|deny icmp
                    source_address source_wildcard_mask
                    destination_address destination_wildcard_mask
                        [icmp_message] [log]
```

exam watch

Remember the information provided in tables 13-6, 13-7, and 13-8: TCP application names and numbers; UDP application names and numbers ; ICMP message types.

Unlike TCP and UDP, ICMP doesn't use ports. Instead, ICMP uses message types. And where TCP and UDP extended ACLs allow you to specify both source and destination ports, ICMP allows you to enter an ICMP message. Table 3-8 shows some of the common ICMP messages and a brief explanation.

You can enter the ICMP message by either its name or its number. Here is a list of message names: **administratively-prohibited**, **alternate-address**, **conversion-error**, **dod-host-prohibited**, **dod-net-prohibited**, **echo**, **echo-reply**, **general-parameter-problem**, **host-isolated**, **host-precedence-unreachable**, **host-redirect**, **host-tos-redirect**, **host-tos-unreachable**, **host-unknown**, **host-unreachable**, **information-reply**, **information-request**, **mask-reply**, **mask-request**, **mobile-redirect**, **net-redirect**, **net-tos-redirect**, **net-tos-unreachable**, **net-unreachable**, **network-unknown**, **no-room-for-option**, **option-missing**, **packet-too-big**, **parameter-problem**, **port-unreachable**, **precedence-unreachable**, **protocol-unreachable**, **reassembly-timeout**, **redirect**, **router-**

TABLE 13-8 Common ICMP Messages

Message Type	Message Description
administratively-prohibited	Message that says that someone filtered a packet
echo	Used by ping to check a destination
echo-reply	Is a response to an echo message created by ping
host-unreachable	The subnet is reachable, but the host is not responding
net-unreachable	The network/subnet is not reachable
traceroute	Filters on traceroute information

advertisement, **router-solicitation**, **source-quench**, **source-route-failed**, **time-exceeded**, **timestamp-reply**, **timestamp-request**, **traceroute**, **ttl-exceeded**, and **unreachable**. If you omit the ICMP message type, all message types are included.

13.04. The CD contains a multimedia demonstration of configuring an extended numbered ACL to permit ICMP traffic through a router.

Activating an Extended IP ACL

Once you have created your extended numbered IP ACL, you must activate it on your router's interface with the following configuration:

```
Router(config)# interface type [module_#]port_#
Router(config-if)# ip access-group ACL_# in|out
```

Note that this is the *same* configuration used with a standard ACL. Once you activate the ACL, the router will begin filtering traffic on the interface.

Extended IP ACL Example 1

Now that you have seen the syntax for creating extended numbered IP ACLs, let's take a look at a couple of configuration examples. Here's the first example:

```
Router(config)# access-list 100 permit tcp
                        any 172.16.0.0 0.0.255.255
                        established log
Router(config)# access-list 100 permit udp
                        any host 172.16.1.1 eq dns log
Router(config)# access-list 100 permit tcp
                        172.17.0.0 0.0.255.255
                        host 172.16.1.2 eq telnet log
Router(config)# access-list 100 permit icmp
                        any 172.16.0.0 0.0.255.255
                        echo-reply log
Router(config)# access-list 100 deny ip any any log
Router(config)# interface ethernet 0
Router(config-if)# ip access-group 100 in
```

The assumption behind this example is that it is restricting what traffic can come into a network. The first statement says that if any TCP session has any source address and is destined to 172.16.0.0/16, it will be permitted if the RST/ACK bits are set (**established**) in the TCP segment header. Remember that the keyword **any** is the same as 0.0.0.0 255.255.255.255. Also, the **log** keyword will cause a match on this statement to be printed on the console. Since a TCP port isn't specified, all TCP connections will match on this statement.

The second line of this example allows a DNS query from any source device to be sent to an internal DNS server (172.16.1.1). Remember that the 0.0.0.0 wildcard mask is removed and the keyword **`host`** is inserted in the front of the IP address. A match on this statement is also logged.

The third line allows any telnet connection from devices in the 172.17.0.0/16 network if the destination device is 172.16.1.2. Remember that telnet uses TCP. A match on this statement is also logged.

The fourth line allows any replies to ping to come back to devices with an address of 172.16.0.0/16. Note that only the echo replies are allowed—echoes are not allowed. A match on this statement is also logged.

The fifth line isn't necessary because all traffic not matching on the previous **`permit`** statements will be dropped. However, if you want to log what is dropped, you'll need to configure this statement with the **`log`** parameter, as is shown in the example. The last part of the configuration shows the ACL applied inbound on `ethernet0`.

Extended IP ACL Example 2

Here's a second extended numbered IP ACL configuration:

```
Router(config)# access-list 101 permit tcp
                        host 199.199.199.1
                        host 200.200.200.1 eq dns
Router(config)# access-list 101 permit udp
                        any host 200.200.200.1 eq dns
Router(config)# access-list 101 permit tcp
                        any host 200.200.200.2 eq www
Router(config)# access-list 101 permit icmp
                        any 200.200.200.0 0.0.0.255
Router(config)# access-list 101 permit tcp
                        any host 200.200.200.3 eq smtp
Router(config)# access-list 101 permit udp
                        host 201.201.201.2
                        host 201.201.201.1 eq rip
Router(config)# interface ethernet 0
Router(config-if)# ip address 201.201.201.1 255.255.255.0
Router(config-if)# ip access-group 100 in
```

The assumption behind this example is that it is restricting traffic as it comes into the network. The first ACL statement allows DNS zone transfers (done via TCP) from 199.199.199.1 to 200.200.200.1; 200.200.200.1, which is on the inside of this network.

The second ACL statement allows DNS queries from anyone to the internal DNS server (200.200.200.1). The third statement allows web traffic from any source to the internal web server (200.200.200.2). The fourth statement allows any ICMP traffic to the 200.200.200.0/24 internal network. The fifth statement allows anyone to send e-mail to the internal email (SMTP) server at 200.200.200.3. The fifth line allows RIP updates to be received from 201.201.201.2, which is a neighboring router off of the `ethernet0` interface. The last part of the configuration activates the ACL on the `ethernet0` interface.

exam watch

Go back and look at these examples again and make sure you understand how they function. Understanding the configuration, activation, verification, and operation of ACLs is very important.

Named ACLs

Starting with IOS 11.2, Cisco routers support both numbered and named ACLs. One of the original limitations of numbered ACLS was that you could create only so many of them. Originally, you could have only 99 standard IP ACLs and 100 extended IP ACLs. The additional numbers weren't added until recently. Starting with IOS 11.2, Cisco allowed you to use names to reference your ACLs instead of, or in combination with, numbered ACLs.

Named ACLs support both the IP and IPX protocols. Unlike in numbered ACLs, in named ACLs you *can* delete a single entry in the ACL. However, there is currently no ability to modify an existing entry or insert a new entry into the middle of an existing ACL. Therefore, you will still need to use the process described earlier in this chapter, in the section "Editing Entries."

Creating Named ACLs

To create a named IP ACL, use the following command:

```
Router(config)# ip access-list standard|extended ACL_name
```

The first thing you must specify is the type of ACL: standard or extended. Second, you must give the ACL a name that groups the ACL statements together. This name must be unique among all named ACLs. Once you enter this command, you are taken into the appropriate *ACL Subconfiguration* mode, as is shown here:

```
Router(config-std-acl)#
-or-
Router(config-ext-acl)#
```

Once you are in the *Subconfiguration* mode, you can enter your ACL commands. For a standard named ACL, use the following configuration:

```
Router(config)# ip access-list standard ACL_name
Router(config-std-acl)# permit|deny source_IP_address
                        [wildcard_mask]
```

For an extended named ACL, use the following configuration:

```
Router(config)# ip access-list extended ACL_name
Router(config-ext-acl)# permit|deny IP_protocol
                        source_IP_address wildcard_mask
                             [protocol_information]
                        destination_IP_address wildcard_mask
                             [protocol_information] [log]
```

As you can see, creating a standard or extended named IP ACL is similar to creating a numbered one.

exam watch

Be familiar with how to create a named ACL and the two different Subconfiguration modes you are taken into depending on whether the ACL is standard or extended.

Activating a Named ACL

Once you have created your named ACL, you need to activate it on your router's interface with the following configuration:

```
Router(config)# interface type [module_#]port_#
Router(config-if)# ip access-group ACL_name in|out
```

The only difference between activating a named versus a numbered ACL on an interface is that you specify the name of the ACL instead of the number.

Example of a Named Access List

In this example, I'll convert the extended IP numbered ACL from the section "Extended IP ACL Example 1" earlier in this chapter. Here's the named version of this ACL:

```
Router(config)# ip access-list extended do_not_enter
Router(config-ext-acl)# permit tcp
                          any 172.16.0.0 0.0.255.255
                          established log
Router(config-ext-acl)# permit udp
                          any host 172.16.1.1 eq dns log
Router(config-ext-acl)# permit tcp
                          172.17.0.0 0.0.255.255
                          host 176.16.1.2 eq telnet log
Router(config-ext-acl)# permit icmp
                          any 176.16.0.0 0.0.255.255
                          echo-reply log
Router(config-ext-acl)# deny ip any any log
Router(config)# interface ethernet 0
Router(config-if)# ip access-group do_not_enter in
```

Both this example and the numbered example do the *exact same thing*. Therefore, it is a matter of personal preference whether you use a named or numbered ACL. My preference is to use numbered ACLs, if only because I've been using them for ten years.

CertCam

13.05. The CD contains a multimedia demonstration of configuring a named IP ACL on a router.

exam watch

You should be very familiar with the syntax of all ACL statements, especially how to configure them and how to evaluate them. Therefore, make sure you really study your ACLs!

Access List Verification

Once you have created and activated your ACLs, you can verify their configuration and operation with various **show** commands. One common command that you can use is the *Privilege EXEC* **show** `running-config` command, which will display your ACL and which interface or interfaces it is activated on. However, there are many other commands you can also use.

If all you want to see is which ACLs are activated on your router's interfaces, you can use the **show** `ip interfaces` command:

```
Router# show ip interfaces
Ethernet0 is up, line protocol is up
  Internet address is 172.16.1.1/24
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is 100
  Proxy ARP is enabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
<-- output omitted -->
```

From the output of this command, you can see that ACL 100, an extended numbered IP ACL, is applied inbound on `ethernet0`.

CertCam

13.06. The CD contains a multimedia demonstration of using the `show ip interfaces` command on a router to verify the activation of your ACLs.

To view the statements in your ACLs, use either of the following two commands:

```
Router# show access-lists [ACL_#_or_name]
Router# show ip access-list [ACL_#_or_name]
```

Here is an example of the **show** `access-lists` command:

```
Router# show access-lists
Extended IP access list 100
    permit tcp 172.16.0.0 0.0.255.255 any established
          (189 matches)
    permit udp host 172.16.1.39 any eq domain
          (32 matches)
    permit icmp host 199.199.199.1 any
IPX sap access list 1000
 deny FFFFFFFF 7
 permit FFFFFFFF 0
```

First, notice that the router keeps track of matches on each statement. The first statement in ACL 100 has had 189 matches against it. You can clear these counters with this command:

```
Router# clear access-list counters [ACL_#_or_name]
```

Also notice that using the **show** `access-lists` command displays all ACLs from all protocols on your router. From the preceding output, there are two ACLs: an extended numbered IP ACL and an IPX SAP ACL. If you want to view only ACLs for IP, use the following command:

```
Router# show ip access-list
Extended IP access list 100
    permit tcp 172.16.0.0 0.0.255.255 any established
         (189 matches)
    permit udp host 172.16.1.39 any eq domain
         (32 matches)
    permit icmp host 199.199.199.1 any
```

If you want to view only a particular ACL, use either of the following two commands:

```
Router# show access-lists 100
Extended IP access list 100
    permit tcp 172.16.0.0 0.0.255.255 any established
         (189 matches)
    permit udp host 172.16.1.39 any eq domain
         (32 matches)
    permit icmp host 199.199.199.1 any
-or-
Router# show ip access-list 100
Extended IP access list 100
    permit tcp 172.16.0.0 0.0.255.255 any established
         (189 matches)
    permit udp host 172.16.1.39 any eq domain
         (32 matches)
    permit icmp host 199.199.199.1 any
```

CertCam

13.07. The CD contains a multimedia demonstration of using the `show [ip] access-list` command on a router to verify the activation of your ACLs.

exam watch

Use the `show ip interfaces` command to see whether or not an IP ACL is applied to your router's interfaces. Use the `show access-lists` command to view all of the ACLs on your router. The `show ip access-list` lists only the IP ACLs on your router. You can always qualify your output by specifying the ACL number in the `show` command.

CERTIFICATION OBJECTIVE 13.05

Placement of ACLs

This section covers design issues with ACLs: where you should place ACLs of a given type (standard or extended). In other words, given the source and destination that you are filtering, on what router and what interface on that router should you activate your ACL? This section covers some of the important points you should be aware of when determining where to put your ACLs.

First, don't go crazy with ACLs and create dozens and dozens of them across all of your routers. This makes testing and troubleshooting your filtering rules almost impossible. If you have followed Cisco's three-layer hierarchy—core, distribution, and access—you'll want to put your ACLs on your distribution layer routers.

The second point to make is that you will want to limit the number of statements in your ACL. An ACL with hundreds of statements is almost impossible to test and troubleshoot. As an example, I had a student in one of the router classes I taught who had a question on an ACL they used at their site—it was six pages long! After I sat with this student, we were able to reduce this to about a page and a half. The original ACL had a lot of unnecessary and overlapping commands that we removed or changed.

As to where you should place your ACLs, the following two rules hold true in most situations:

- Standard ACLs should be placed as close to the *destination* devices as possible.
- Extended ACLs should be placed as close to the *source* devices as possible.

exam watch

Remember the two rules as to where you should apply your ACLs.

Standard ACLs

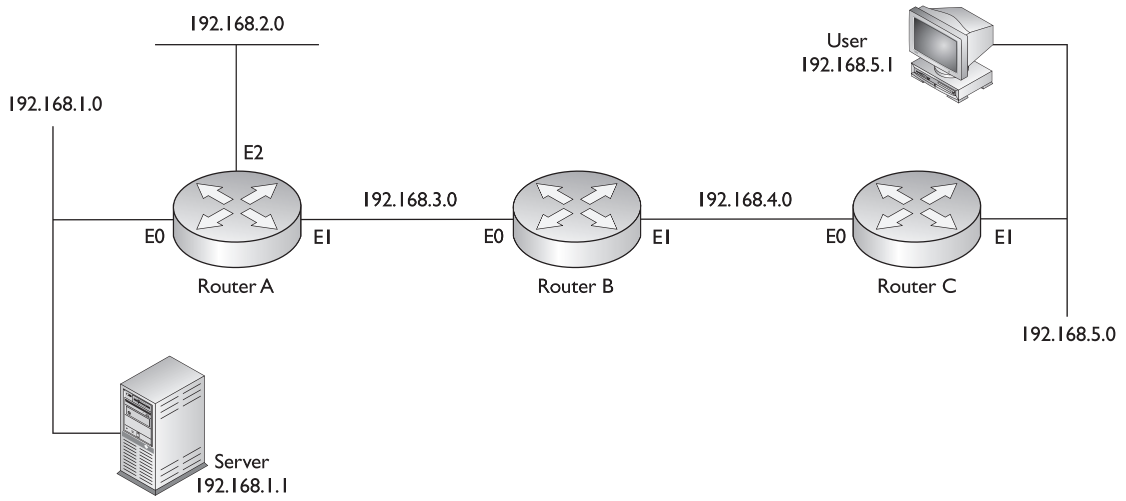
You want to place standard ACLs as close to the *destination* that you want to prevent the source from reaching, since they allow you to filter only on the source IP address in the packet headers. If you put the standard ACL too close to the source, then you could be preventing the source from accessing other valid services in your network. By putting the standard ACL as close to the destination as possible, you are still allowing the source to access other resources, while restricting it from accessing the remote destination device or devices.

Let's take a look at an example to illustrate the placement of standard ACLs. I'll use the network shown in Figure 13-1. In this example, the user (192.168.5.1) should be prevented from accessing the server (192.168.1.1). Here is the ACL configuration:

```
Router(config)# access-list 1 deny 192.168.5.1 0.0.0.0
Router(config)# access-list 1 permit any
```

As you can see from this example, the goal is to prevent 192.168.5.1 from accessing 192.168.1.1, but to allow everyone else to access it. Let's discuss the options as to where you can place this ACL. Your first choice is to place this ACL on RouterC. If you placed it here, 192.168.5.1 would not be able to reach 192.168.1.1, but the user wouldn't be able to access anything else either. If you placed the ACL on RouterB, the user would be able to access the 192.168.4.0 network, but nothing

FIGURE 13-1 Placement of ACLs



else. You actually have two choices for placing the ACL on RouterA: interfaces `E0` and `E1`. If you placed it inbound on `E1`, then the user wouldn't be able to access network 192.168.2.0. Therefore, you would have to place it outbound on `E0` of RouterA.

Note that there is still an issue with using standard ACLs—any traffic from 192.168.5.1 is dropped as it attempts to leave this interface. So, the user is prevented from reaching not only the server but anything else on this segment. Another issue with standard ACLs, since you typically place them as close to the destination as possible, is that they are not very network-friendly: packets travel almost all of the way to the destination and *then* they are dropped. This wastes bandwidth in your network, especially if the source is sending a lot of traffic to the destination.

Extended ACLs

Given the preceding example, it would be much better to place the standard ACL as close to the source as possible to prevent unwanted traffic from traversing almost the whole network before being dropped. With a standard ACL, though, you would be preventing the user from accessing most of the resources in the network.

Extended ACLs, however, don't have this limitation, since they can filter on *both* the source and destination addresses in the IP packet headers. Given this ability, it is recommended that you place extended ACLs as *close* to the source as possible, thus preventing unwanted traffic from traversing your network. With an extended ACL, since you can filter on both addresses, you can prevent a source from accessing a particular destination or destinations but still allow it to access others.

With our preceding example, your configuration would look like this when using an extended ACL:

```
Router(config)# access-list 100 deny ip host 192.168.5.1
                                        host 192.168.1.1
Router(config)# access-list 100 permit ip any any
```

This configuration example is preventing only traffic from 192.168.5.1 to 192.168.1.1. Now the question is, where you should place this ACL? Again, you want to put this ACL as close to the source as possible. This means that you should place it on RouterC. RouterC has two interfaces, though. Again, remember that it should be placed as close to the *source* as possible. This means that the ACL should be placed on RouterC's `E1` interface. If you were to place it on `E0`, and the router had another interface that it could use to reach the destination, the source still might be able to get around the filter. If you place it on RouterC's `E1` interface, 192.168.5.1 can access every location except 192.168.1.1. Likewise, any other traffic is permitted to go anywhere in the network.

You can be more specific with your filtering in this example. For example, if you want to restrict just telnet access, but allow other types of access from 192.168.5.1 to 192.168.1.1, then you should specify the IP protocol (**`tcp`**) and the destination port name or number (**`telnet`** or **`23`**).

CERTIFICATION SUMMARY

ACLs can be used to filter traffic and routing information, restrict telnets to your router, prioritize traffic, trigger DDR phone calls, and many other things. ACLs are statements grouped together by a number of names that define traffic that should be permitted or denied. ACLs can be applied in either the inbound or outbound direction. With an inbound ACL, the ACL is processed first before any other processing is done on the packet. With an outbound ACL, the packet is routed to the outbound interface first and then the ACL is processed.

Standard IP ACLs allow you to filter on the source IP address, while extended IP ACLs allow you to filter on the source and destination IP addresses, the IP protocol, and protocol information (such as port numbers). ACLs are processed top-down until a match is found; at that point, no other statements are processed. Therefore, the order of the statements is important. If no match is found, the implicit deny rule takes place and the packet is dropped. You can have one ACL, per protocol, per interface, per direction on that interface. There are two special filtering rules for ACLs: you cannot filter traffic the router itself originates, and applying an empty ACL to an interface permits all traffic by default.

Standard ACLs can have numbers ranging 1–99 and 1300–1999, and extended ACLs can have numbers ranging 100–199 and 2000–2699. Standard ACLs should be placed as close to the destination as possible, while extended ACLs should be placed as close to the source as possible.

To create a numbered ACL, use the **`access-list`** command. Use the **`ip access-group`** command to activate your ACL on an interface. To filter telnet traffic to and from your router, activate the standard IP ACL on your VTY lines with the **`access-class`** command. When making changes to ACLs, paste the ACL configuration into a text editor, make your changes, remove the application of the ACL on the router's interface(s), delete the ACL, paste the text editor ACL into your router, and reactivate it. To create a named ACL, use the **`ip access-list standard|extended`** command. This will take you into the appropriate *Subconfiguration* mode.

Wildcard masks allow you to match a single address, a range of addresses, or all addresses. Basically, a wildcard mask is like an inverted subnet mask. A 0 in a bit position means match, and a 1 means ignore. To convert a subnet mask to a wildcard mask, subtract each octet in the subnet mask from 255, resulting in the corresponding octet value for the wildcard mask. With standard ACLs, if you omit the wildcard mask, it defaults to 0.0.0.0.

The **`established`** keyword in an extended ACL allows you to look at the TCP flags to determine whether or not to allow the packets. The **`log`** keyword will display matches of all matched packets.

The **`show ip interfaces`** command will display any ACLs that have been activated on your router's interfaces. The **`show access-lists`** command displays all ACLs configured on your router for all protocols. The **`show ip access-list`** command displays only IP ACLs.

✓ TWO-MINUTE DRILL

ACL Overview

❑ ACLs allow you to filter traffic, restrict telnets to the router, filter routing information, prioritize WAN traffic, trigger dialup connections, change administrative distances of routes, and many other things.

❑ ACLs can be created using either numbers or names. There are two basic types: standard and extended. Standard ACLs allow you to filter only the source IP address, whereas extended IP ACLs allow you to filter on source and destination addresses, IP protocols, and protocol information.

❑ There are two actions the router can take when a match is found on an ACL: permit or deny. ACLs are processed top-down, where the order is important. Upon the first match, no other statements are processed. There is an implicit deny at the end of the list. You cannot filter traffic the router itself originates. When adding ACL statements, note that they are always added to the bottom. Only named ACLs allow you to delete a specific entry.

Basic ACL Configuration

❑ The **`access-list`** command creates an ACL and the **`ip access-group`** command activates the ACL on an interface. You can filter traffic as it enters (**`in`**) or leaves (**`out`**) an interface. To delete a complete access control list, use the **`no access-list`** command, followed by its number.

❑ Standard IP ACLs use numbers in the ranges 1–99 and 1300–1999, and extended IP ACLs use list numbers 100–199 and 2000–2699.

Wildcard Masks

❑ A wildcard mask is like an inverted subnet mask. A 0 in a bit position of the wildcard mask means the corresponding bit position in the condition's address must match that in the IP packet. A 1 in a bit position of the wildcard mask means there doesn't have to be a match.

❑ A wildcard mask of 0.0.0.0 means that the entire address must match. Precede the word **`host`** before an address accomplishes the same thing. A wildcard mask of 255.255.255.255 indicates that any address matches: you can replace the address and wildcard mask with the keyword **`any`**.

❑ To invert a subnet mask into a wildcard mask, subtract each octet in the subnet mask from 255, which will result in the corresponding octet value for the wildcard mask.

Types of ACLs

❑ Standard ACLs can filter only on the source IP address. If you omit the wildcard mask, it defaults to 0.0.0.0. Use the **`access-class`** command to activate a standard ACL to restrict telnet access to a router.

❑ Use the **`terminal monitor`** command to view console output on nonconsole connections, such as VTYs.

❑ Extended IP ACLs allow you to filter on both the source and destination IP addresses (where you must specify the wildcard mask for both), the IP protocol (TCP, UDP, ICMP, and so on), and protocol information (such as ICMP message types or TCP and UDP source and destination port numbers). If you want to match on all IP traffic, use the keyword **`ip`** for the protocol parameter.

❑ Use the **`ip access-list standard|extended`** *`ACL_name`* command to create a named ACL. This takes you into the *ACL Subconfiguration* mode.

❑ The **`show running-config`** command will display your configured ACLs and the interfaces they are activated on. The **`show ip interfaces`** command shows the ACLs activated on a router's interfaces. The **`show [ip] access-lists`** command displays the statements in a router's ACLs.

Placement of ACLs

❑ Standard ACLs should be placed as close to the *destination* devices as possible.

❑ Extended ACLs should be placed as close to the *source* devices as possible.

SELF TEST

The following Self Test questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully, as there may be more than one correct answer. Choose all correct answers for each question.

ACL Overview

1. Which of the following are not features of ACLs?

A. Restricting telnet access to a router
B. Prioritizing WAN traffic
C. Filtering traffic from the router
D. Triggering dialup phone calls

2. Which of the following is true concerning ACLs?

A. Order of the statements is automatic.
B. All statements are processed.
C. If no match is found, the packet is permitted.
D. You can delete a specific statement in a named list.

3. The last statement in an ACL is called the __________ ____________ statement.

Basic ACL Configuration

4. There are ___________ actions a router can take when there is a match on an ACL statement (enter a number).

5. Which command activates an IP ACL on a router's interface?

A. **`access-list`**
B. **`ip access-group`**
C. **`access-class`**
D. **`access-group`**

Wildcard Masks

6. A ___________ in a bit position of a wildcard mask means that the same bit position in the condition address must match the same bit position in the wildcard mask in order to execute the ACL's action.

A. 0
B. 1

7. Enter the wildcard mask value to match on every bit position in an address: __________.
8. Enter the wildcard mask value for the subnet mask of 255.255.248.0: ___________.

Types of ACLs

9. Choose the following that a standard IP ACL can match on.
   A. Destination address
   B. IP protocol
   C. IP protocol information
   D. None of the above
10. Enter the router command to view console output on a non-console line connection: __________.
11. Enter the standard IP ACL command to permit traffic from 192.168.1.0/24, using a list number of 10: ___________.
12. Enter the extended IP ACL command to permit all ICMP traffic from 172.16.0.0/16 to 172.17.0.0/17, using a list number of 101: ___________.
13. Which router command creates a standard named ACL called *test*?
   A. **`ip access-list test`**
   B. **`access-list test`**
   C. **`ip access-list standard test`**
   D. **`access-list standard test`**
14. Enter the router command to activate an ACL with a name of *test* inbound on an interface: ___________.
15. Enter the router command to delete all of the statements in access-list 100: __________.

Placement of ACLs

16. Extended IP ACLs should be placed as close to the ___________ device as possible.
   A. Source
   B. Destination

SELF TEST ANSWERS

ACL Overview

1. ☑ **C.** ACLs cannot filter traffic the router originates, such as pings or traceroutes.
   ☒ **A**, **B**, and **C** are ACL features.

2. ☑ **D.** You can delete a specific ACL statement in a named ACL, but not a numbered ACL.
   ☒ **A** is not true because all statements are always added at the bottom of the ACL. **B** is not true because as soon as there is a statement match, no more statements are processed. **C** is not true because the implicit deny at the end of every ACL drops a non-matching packet.

3. ☑ The last statement in an ACL is called an *implicit deny* statement: this statement, which cannot be seen with any **`show`** commands, causes the router to drop any packet that doesn't match any preceding ACL statements.

Basic ACL Configuration

4. ☑ There are **2** actions a router can take when there is a match on an ACL statement: **`permit`** or **`deny`**.

5. ☑ **B.** The **`ip access-group`** command activates an ACL on a router's interface.
   ☒ **A** is incorrect because it creates an ACL statement in a list. **C** is incorrect because it activates a standard ACL on a line, not an interface. **D** is a nonexistent command.

Wildcard Masks

6. ☑ **A.** A 0 in a bit position of a wildcard mask means that the same bit position in the condition address must match that bit position in the wildcard mask in order to execute the ACL's action.
   ☒ **B** means that the bit position doesn't have to match.

7. ☑ The value **`0.0.0.0`** is a wildcard mask that says to match on every bit position in an address.

8. ☑ The inverted subnet mask for 255.255.248.0 is **`0.0.7.255`**. The trick is to subtract the subnet mask octets from 255.

Types of ACLs

9. ☑ **D**. Standard IP ACLs can match only on source IP addresses.
   ☒ **A**, **B**, and **C** are things extended IP ACLs can match on.

10. ☑ Use the **`terminal monitor`** command to view console output on a nonconsole line connection, such as a VTY or an auxiliary line.

11. ☑ Enter the standard IP ACL command to permit traffic from 192.168.1.0/24, using a list number of 10: **`access-list 10 permit 192.168.1.0 0.0.0.255`**.

12. ☑ Enter the extended IP ACL command to permit all ICMP traffic from 172.16.0.0/16 to 172.17.0.0/17, using a list number of 101: **`access-list 101 permit icmp 172.16.0.0 0.0.255.255 172.17.0.0 0.0.127.255`**. Notice the subnet mask value for 172.17.0.0, which is 17 bits!

13. ☑ **C.** The **`ip access-list standard test`** command creates a standard ACL called *test*.
    ☒ **A**, **B**, and **D** are invalid commands.

14. ☑ Enter the router command to activate an access list with a name of *test* inbound on an interface: **`ip access-group test in`**.

15. ☑ Enter the router command to delete access-list 100: **`no access-list 100`**.

Placement of ACLs

16. ☑ **A.** Extended IP ACLs should be placed as close to the source device as possible.
    ☒ **B** is true for standard ACLs.