Security Configuration Guide: Access Control Lists, Cisco IOS Release 15M&T

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

IP Access List Overview

Access control lists (ACLs) perform packet filtering to control which packets move through the network and where. Such control provides security by helping to limit network traffic, restrict the access of users and devices to the network, and prevent traffic from leaving a network. IP access lists can reduce the chance of spoofing and denial-of-service attacks and allow dynamic, temporary user access through a firewall.

IP access lists can also be used for purposes other than security, such as bandwidth control, restricting the content of routing updates, redistributing routes, triggering dial-on-demand (DDR) calls, limiting debug output, and identifying or classifying traffic for quality of service (QoS) features. This module provides an overview of IP access lists.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About IP Access Lists

Benefits of IP Access Lists

Access control lists (ACLs) perform packet filtering to control the flow of packets through a network. Packet filtering can restrict the access of users and devices to a network, providing a measure of security. Access lists can save network resources by reducing traffic. The benefits of using access lists are as follows:

- Authenticate incoming rsh and rcp requests—Access lists can simplify the identification of local users, remote hosts, and remote users in an authentication database that is configured to control access to a device. The authentication database enables Cisco software to receive incoming remote shell (rsh) and remote copy (rcp) protocol requests.
- Block unwanted traffic or users—Access lists can filter incoming or outgoing packets on an interface, thereby controlling access to a network based on source addresses, destination addresses, or user authentication. You can also use access lists to determine the types of traffic that are forwarded or blocked at device interfaces. For example, you can use access lists to permit e-mail traffic to be routed through a network and to block all Telnet traffic from entering the network.
- Control access to vty—Access lists on an inbound vty (Telnet) can control who can access the lines to a device. Access lists on an outbound vty can control the destinations that the lines from a device can reach.
- Identify or classify traffic for QoS features—Access lists provide congestion avoidance by setting the IP precedence for Weighted Random Early Detection (WRED) and committed access rate (CAR). Access lists also provide congestion management for class-based weighted fair queueing (CBWFQ), priority queueing, and custom queueing.
- Limit debug command output—Access lists can limit debug output based on an IP address or a protocol.
- Provide bandwidth control—Access lists on a slow link can prevent excess traffic on a network.
- Provide NAT control—Access lists can control which addresses are translated by Network Address Translation (NAT).
- Reduce the chance of DoS attacks—Access lists reduce the chance of denial-of-service (DoS) attacks. Specify IP source addresses to control traffic from hosts, networks, or users from accessing your network. Configure the TCP Intercept feature to can prevent servers from being flooded with requests for connection.
- Restrict the content of routing updates—Access lists can control routing updates that are sent, received, or redistributed in networks.
- Trigger dial-on-demand calls—Access lists can enforce dial and disconnect criteria.

Border Routers and Firewall Routers Should Use Access Lists

There are many reasons to configure access lists; for example, you can use access lists to restrict contents of routing updates or to provide traffic flow control. One of the most important reasons to configure access lists is to provide a basic level of security for your network by controlling access to it. If you do not configure access lists on your router, all packets passing through the router could be allowed onto all parts of your network.

An access list can allow one host to access a part of your network and prevent another host from accessing the same area. In the figure below, by applying an appropriate access list to the interfaces of the router, Host A is allowed to access the Human Resources network and Host B is prevented from accessing the Human Resources network.

Access lists should be used in firewall routers, which are often positioned between your internal network and an external network such as the Internet. You can also use access lists on a router positioned between two parts of your network, to control traffic entering or exiting a specific part of your internal network.

To provide some security benefits of access lists, you should at least configure access lists on border routers--routers located at the edges of your networks. Such an access list provides a basic buffer from the outside network or from a less controlled area of your own network into a more sensitive area of your network. On these border routers, you should configure access lists for each network protocol configured on the router interfaces. You can configure access lists so that inbound traffic or outbound traffic or both are filtered on an interface.

Access lists are defined on a per-protocol basis. In other words, you should define access lists for every protocol enabled on an interface if you want to control traffic flow for that protocol.

Definition of an Access List

An access list is a sequential list consisting of at least one **permit** statement and possibly one or more **deny** statements. In the case of IP access lists, the statements can apply to IP addresses, upper-layer IP protocols, or other fields in IP packets. The access list is identified and referenced by a name or a number. The access list acts as a packet filter, filtering packets based on the criteria defined in the access list.

An access list may be configured, but it does not take effect until the access list is either applied to an interface (with the **ip access-group** command), a virtual terminal line (vty) (with the **access-class**command), or referenced by some other command that accepts an access list. Access lists have many uses, and therefore many Cisco IOS software commands accept a reference to an access list in their command syntax. Multiple commands can reference the same access list.

In the following configuration excerpt, the first three lines are an example of an IP access list named branchoffices, which is applied to serial interface 0 on incoming packets. No sources other than those on the networks specified by each source address and mask pair can access this interface. The destinations for packets coming from sources on network 172.20.7.0 are unrestricted. The destination for packets coming from sources on network 172.29.2.0 must be 172.25.5.4.

```
ip access-list extended branchoffices
 10 permit 172.20.7.0 0.0.0.3 any
 20 permit 172.29.2.0 0.0.0.255 host 172.25.5.4
!
interface serial 0
 ip access-group branchoffices in
```

Software Processing of an Access List

The following general steps describe how the Cisco IOS software processes an access list when it is applied to an interface, a vty, or referenced by some other Cisco IOS command. These steps apply to an access list that has 13 or fewer access list entries.

- The software receives an IP packet and tests parts of each packet being filtered against the conditions in the access list, one condition (**permit** or **deny** statement) at a time. For example, the software tests the source and destination addresses of the packet against the source and destination addresses in a **permit** or **deny**statement.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the rest of the statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies a packet, the software discards the packet and returns an ICMP Host Unreachable message.
- If no conditions match, the software drops the packet. This is because each access list ends with an unwritten, implicit **deny** statement. That is, if the packet has not been permitted by the time it was tested against each statement, it is denied.

In later Cisco IOS releases such as Release 12.4, 12.2S, and 12.0S, by default, an access list that has more than 13 access list entries is processed differently from one that has 13 or fewer entries. In order to be more efficient, an access list with more than 13 entries is processed using a trie-based lookup algorithm. This process will happen automatically; it does not need to be configured.

Access List Rules

The following rules apply to access lists:

- Only one access list per interface, per protocol, and per direction is allowed.
- An access list must contain at least one **permit** statement or all packets are denied entry into the network.
- The order in which access list conditions or match criteria are configured is important. While deciding whether to forward or block a packet, Cisco software tests the packet against each criteria statement in the order in which these statements are created. After a match is found, no more criteria statements are checked. The same **permit** or **deny** statements specified in a different order can result in a packet being passed under one circumstance and denied in another circumstance.
- If an access list is referenced by a name, but the access list does not exist, all packets pass. An interface or command with an empty access list applied to it permits all traffic into the network.
- Standard access lists and extended access lists cannot have the same name.
- Inbound access lists process packets before the packets are routed to an outbound interface. Inbound access lists that have filtering criteria that deny packet access to a network saves the overhead of routing lookup. Packets that are permitted access to a network based on the configured filtering criteria are processed for routing. For inbound access lists, when you configure a **permit** statement, packets are processed after they are received, and when you configure a **deny** statement, packets are discarded.
- Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed by the outbound access list. For outbound access lists, when you configure a **permit** statement, packets are sent to the output buffer, and when you configure a **deny** statement, packets are discarded.

- An access list can control traffic arriving at a device or leaving a device, but not traffic originating at a device.

Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient, useful access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list**command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.
- While you are creating an access list or after it is created, you might want to delete an entry.
    - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.
    - You can delete an entry from a named access list. Use the **no permit**or **no deny** command to delete the appropriate entry.
- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.
- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

Named or Numbered Access Lists

All access lists must be identified by a name or a number. Named and numbered access lists have different command syntax. Named access lists are compatible with Cisco IOS Release 11.2 and later. Named access lists are more convenient than numbered access lists because you can specify a meaningful name that is easier to remember and associate with a purpose. You may reorder statements in or add statements to a named access list.

Named access list are newer than numbered access lists and support the following features that are not supported in numbered access lists:

- TCP flag filtering
- IP option filtering
- noncontiguous ports
- reflexive access lists
- ability to delete entries with the **no permit** or **no deny** command

Not all commands that accept a numbered access list will accept a named access list. For example, virtual terminal lines use only numbered access lists.

Standard or Extended Access Lists

All access lists are either standard or extended access lists. If you only intend to filter on a source address, the simpler standard access list is sufficient. For filtering on anything other than a source address, an extended access list is necessary.

- Named access lists are specified as standard or extended based on the keyword **standard** or **extended** in the **ip access-list** command syntax.
- Numbered access lists are specified as standard or extended based on their number in the **access-list** command syntax. Standard IP access lists are numbered 1 to 99 or 1300 to 1999; extended IP access lists are numbered 100 to 199 or 2000 to 2699. The range of standard IP access lists was initially only 1 to 99, and was subsequently expanded with the range 1300 to 1999 (the intervening numbers were assigned to other protocols). The extended access list range was similarly expanded.

Standard Access Lists

Standard IP access lists test only source addresses of packets (except for two exceptions). Because standard access lists test source addresses, they are very efficient at blocking traffic close to a destination. There are two exceptions when the address in a standard access list is not a source address:

- On outbound VTY access lists, when someone is trying to telnet, the address in the access list entry is used as a destination address rather than a source address.
- When filtering routes, you are filtering the network being advertised to you rather than a source address.

Extended Access Lists

Extended access lists are good for blocking traffic anywhere. Extended access lists test source and destination addresses and other IP packet data, such as protocols, TCP or UDP port numbers, type of service (ToS), precedence, TCP flags, IP options, and TTL value. Extended access lists can also provide capabilities that standard access lists cannot, such as the following:

- Filtering IP Options
- Filtering TCP flags
- Filtering noninitial fragments of packets (see the module “Refining an IP Access List”)
- Time-based entries (see "Time-Based and Distributed Time-Based Access Lists" and the module “Refining an IP Access List”)
- Dynamic access lists (see the section "Types of IP Access Lists")
- Reflexive access lists (see the section "Types of IP Access Lists" and the module “Configuring IP Session Filtering [Reflexive Access Lists])

Note Packets that are subject to an extended access list will not be autonomous switched.

IP Packet Fields You Can Filter to Control Access

You can use an extended access list to filter on any of the following fields in an IP packet. Source address and destination address are the two most frequently specified fields on which to base an access list:

- Source address--Specifies a source address to control packets coming from certain networking devices or hosts.
- Destination address--Specifies a destination address to control packets being sent to certain networking devices or hosts.
- Protocol--Specifies an IP protocol indicated by the keyword **eigrp**, **gre**, **icmp**, **igmp**, **ip**, **ipinip**, **nos**, **ospf**, **tcp**, or **udp**, or indicated by an integer in the range from 0 to 255 (representing an Internet protocol). If you specify a transport layer protocol (**icmp**, **igmp**, **tcp**, or **udp**), the command has a specific syntax.
    - Ports and non-contiguous ports--Specifies TCP or UDP ports by a port name or port number. The port numbers can be noncontiguous port numbers. Port numbers can be useful to filter Telnet traffic or HTTP traffic, for example.
    - TCP flags--Specifies that packets match any flag or all flags set in TCP packets. Filtering on specific TCP flags can help prevent false synchronization packets.
- IP options--Specifies IP options; one reason to filter on IP options is to prevent routers from being saturated with spurious packets containing them.

Wildcard Mask for Addresses in an Access List

Address filtering uses wildcard masking to indicate to the software whether to check or ignore corresponding IP address bits when comparing the address bits in an access list entry to a packet being submitted to the access list. By carefully setting wildcard masks, you can specify one or more IP addresses for permit or deny tests.

Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.

- A wildcard mask bit 0 means check the corresponding bit value; they must match.
- A wildcard mask bit 1 means ignore that corresponding bit value; they need not match.

If you do not supply a wildcard mask with a source or destination address in an access list statement, the software assumes an implicit wildcard mask of 0.0.0.0, meaning all values must match.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

The table below shows examples of IP addresses and masks from an access list, along with the corresponding addresses that are considered a match.

Table 1: Sample IP Addresses, Wildcard Masks, and Match Results

Address	Wildcard Mask	Match Results
0.0.0.0	255.255.255.255	All addresses will match the access list conditions.
172.18.0.0/16	0.0.255.255	Network 172.18.0.0
172.18.5.2/16	0.0.0.0	Only host 172.18.5.2 matches
172.18.8.0	0.0.0.7	Only subnet 172.18.8.0/29 matches
172.18.8.8	0.0.0.7	Only subnet 172.18.8.8/29 matches
172.18.8.15	0.0.0.3	Only subnet 172.18.8.15/30 matches
10.1.2.0	0.0.254.255 (noncontiguous bits in mask)	Matches any even-numbered network in the range of 10.1.2.0 to 10.1.254.0

Access List Sequence Numbers

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When you add a new entry, you specify the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Access List Logging

The Cisco IOS software can provide logging messages about packets permitted or denied by a single standard or extended IP access list entry. That is, any packet that matches the entry will cause an informational logging message about the packet to be sent to the console. The level of messages logged to the console is controlled by the **logging console** global configuration command.

The first packet that triggers the access list entry causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged. The logging message includes the access list number, whether the packet was permitted or denied, the source IP address of the packet, and the number of packets from that source permitted or denied in the prior 5-minute interval.

However, you can use the **ip access-list log-update** command to set the number of packets that, when match an access list (and are permitted or denied), cause the system to generate a log message. You might want to do this to receive log messages more frequently than at 5-minute intervals.

Caution If you set the *number-of-matches* argument to 1, a log message is sent right away, rather than caching it; every packet that matches an access list causes a log message. A setting of 1 is not recommended because the volume of log messages could overwhelm the system.

Even if you use the **ip access-list log-update** command, the 5-minute timer remains in effect, so each cache is emptied at the end of 5 minutes, regardless of the count of messages in each cache. Regardless of when the log message is sent, the cache is flushed and the count reset to 0 for that message the same way it is when a threshold is not specified.

Note The logging facility might drop some logging message packets if there are too many to be handled or if there is more than one logging message to be handled in 1 second. This behavior prevents the router from crashing due to too many logging packets. Therefore, the logging facility should not be used as a billing tool or an accurate source of the number of matches to an access list.

Alternative to Access List Logging

Packets matching an entry in an ACL with a log option are process switched. It is not recommended to use the log option on ACLs, but rather use NetFlow export and match on a destination interface of Null0. This is done in the CEF path. The destination interface of Null0 is set for any packet that is dropped by the ACL.

Additional IP Access List Features

Beyond the basic steps to create a standard or extended access list, you can enhance your access lists as mentioned below. Each of these methods is described completely in the module entitled "Refining an Access List."

- You can impose dates and times when **permit** or **deny** statements in an extended access list are in effect, making your access list more granular and specific to an absolute or periodic time period.
- After you create a named access list, you might want to add entries or change the order of the entries, known as resequencing an access list.
- You can achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

Time-Based and Distributed Time-Based Access Lists

Time-based access lists implement access list entries based on particular times of the day or week. This is an advantage when you don't want access list entries always in effect or in effect as soon as they are applied. Use time-based access lists to make the enforcement of permit or deny conditions granular, based on time and date.

Distributed time-based access lists are those that are supported on line cards for the Cisco 7500 series routers. Packets destined for an interface configured with time-based access lists are distributed switched through the line card.

Types of IP Access Lists

There are several types of access lists that are distinct because of how they are triggered, their temporary nature, or how their behavior differs from an ordinary access list.

Authentication Proxy

Authentication proxy provides dynamic, per-user authentication and authorization, authenticating users against industry standard TACACS+ and RADIUS authentication protocols. Authenticating and authorizing connections by users provides more robust protection against network attacks.

Context-Based Access Control

Context-based access control (CBAC) examines not only network layer and transport layer information, but also the application-layer protocol information (such as FTP information) to learn about the state of TCP and UDP connections. CBAC maintains connection state information for individual connections. This state information is used to make intelligent decisions about whether packets should be permitted or denied, and dynamically creates and deletes temporary openings in the firewall.

Dynamic Access Lists with the Lock-and-Key Feature

Dynamic access lists provide temporary access to designated users who are using Telnet to reach designated hosts through a firewall. Dynamic access lists involve user authentication and authorization.

Reflexive Access Lists

Reflexive access lists provide filtering on upper-layer IP protocol sessions. They contain temporary entries that are automatically created when a new IP session begins. They are nested within extended, named IP access lists that are applied to an interface. Reflexive access lists are typically configured on border routers, which pass traffic between an internal and external network. These are often firewall routers. Reflexive access lists do not end with an implicit deny statement because they are nested within an access list and the subsequent statements need to be examined.

Where to Apply an Access List

If you are applying an access list to an interface, carefully consider whether to specify it as **in** (inbound) or **out** (outbound). Applying an access list to an incoming or outgoing interface controls the traffic that will enter or leave the router's interface or process level (in the case of filtering on TTL values).

- When an inbound access list is applied to an interface, after the software receives a packet, the software checks the packet against the access list statements. If the access list permits the packet, the software continues to process the packet. Therefore, filtering on incoming packets can save router resources because filtered packets will not go through the router.
- Access lists that apply to outbound packets are filtering packets that have already gone through the router. Packets that pass the access list are transmitted (sent) out the interface.
- The TCP ACL splitting feature of Rate-Based Satellite Control Protocol (RBSCP) is an example of a feature that can be used on an outgoing interface. The access list controls which packets are subject to TCP ACK splitting.

Access lists can be used in ways other than applying them to interfaces. The following are additional places to apply an access list.

- To restrict incoming and outgoing connections between a particular vty (into a Cisco device) and the network devices at addresses in an access list, apply an access list to a line. See the "Controlling Access to a Virtual Terminal Line" module.
- Referencing an access list from a **debug** command limits the amount of information displayed to only the information permitted by the access list, such as sources, destinations, or protocols, for example.
- Access lists can be used to control routing updates, to control dial-on-demand routing (DDR), and to control quality of service (QoS) features, for example. See the appropriate configuration chapters for using access lists with these features.

Where to Go Next

You must first decide what you want to restrict, and then select the type of access list that achieves your goal. Next, you will create an access list that permits or denies packets based on values in the fields you specify, and finally, you will apply the access list (which determines its placement).

Assuming you have decided what you want to restrict and what type of access list you need, your next step is to create an access list. Creating an access list based on source address, destination address, or protocol is described in the "Creating an IP Access List and Applying It to an Interface" module. You could create an access list that filters on other fields, as described in "Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values." If you want to control access to a virtual line, see "Controlling

Access to a Virtual Terminal Line.” If the purpose of your access list is to control routing updates or QoS features, for example, see the appropriate technology chapter.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IP access list commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS IP Application Services Command Reference*
Filtering on source address, destination address, or protocol	“Creating an IP Access List and Applying It to an Interface”
Filtering on IP Options, TCP flags, noncontiguous ports, or TTL	“Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values”
Restricting access to a vty line.	"Controlling Access to a Virtual Terminal Line"

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for IP Access List Overview

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for IP Access List Overview

Feature Name	Releases	Feature Information
IP Access List Overview	12.0(32)S4 15.4(1)S	Access control lists (ACLs) perform packet filtering to control which packets move through the network and where. Such control provides security by helping to limit network traffic, restrict the access of users and devices to the network, and prevent traffic from leaving a network. IP access lists can reduce the chance of spoofing and denial-of-service attacks and allow dynamic, temporary user access through a firewall. In Cisco IOS Release 15.4(1)S, support was added for the Cisco ASR 901S series router.

CHAPTER 2

Access Control List Overview and Guidelines

Cisco provides basic traffic filtering capabilities with access control lists (also referred to as access lists). You can configure access control lists (ACLs) for all routed network protocols (IP, AppleTalk, and so on) to filter protocol packets when these packets pass through a device. You can configure access lists on your device to control access to a network; access lists can prevent certain traffic from entering or exiting a network. This module provides an overview of access lists.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About Access Control Lists

Overview of an Access Control List

Access lists filter network traffic by controlling the forwarding or blocking of routed packets at the interface of a device. A device examines each packet to determine whether to forward or drop that packet, based on the criteria specified in access lists.

The criteria that can be specified in an access list include the source address of the traffic, the destination address of the traffic, and the upper-layer protocol.

Note Some users might successfully evade basic access lists because these lists require no authentication.

Functions of an Access Control List

There are many reasons to configure access lists; for example, to restrict contents of routing updates or to provide traffic flow control. One of the most important reasons to configure access lists is to provide security for your network, which is the focus of this module.

Use access lists to provide a basic level of security for accessing your network. If you do not configure access lists on your device, all packets passing through the device are allowed access to all parts of your network.

Access lists can allow a host to access a part of your network and prevent another host from accessing the same area. In the figure below, Host A is allowed to access the Human Resources network, but Host B is prevented from accessing the Human Resources network.

You can also use access lists to define the type of traffic that is forwarded or blocked at device interfaces. For example, you can permit e-mail traffic to be routed but at the same time block all Telnet traffic.

Scenarios for Configuring an Access Control List

Access lists should be configured on “firewall” devices, which are often positioned between your internal network and an external network such as the Internet. You can also use access lists on a device positioned between two parts of your network to control traffic entering or exiting a specific part of your internal network.

To use the security benefits of access lists, you should, at the minimum, configure access lists on edge devices. Configuring access lists on edge devices provides a basic buffer from the outside network or from a less controlled area of your own network into a more sensitive area of your network.

On border devices, you should configure access lists for each network protocol that is configured on device interfaces. You can configure access lists so that inbound traffic or outbound traffic or both are filtered on an interface.

Access lists must be defined on a per-protocol basis. In other words, you should define access lists for every protocol enabled on an interface if you want to control traffic flow for those protocols.

Note Some protocols refer to access lists as filters.

Differences Between Basic and Advanced Access Control Lists

This module describes how to use standard and static extended access lists, which are types of basic access lists. A basic access list should be used with each routed protocol that is configured on device interfaces.

Besides basic access lists described in this module, there are also advanced access lists available, which provide additional security features and provide greater control over packet transmission.

Access Control List Configuration

Each protocol has its own set of specific tasks and rules to provide traffic filtering. In general, most protocols require at least two basic steps to be completed. The first step is to create an access list, and the second step is to apply the access list to an interface.

Note Some protocols refer to access lists as filters and to the act of applying the access lists to interfaces as filtering.

Create an Access Control List

Create access lists for each protocol that you wish to filter, per device interface. For some protocols, you can create one access list to filter inbound traffic and another access list to filter outbound traffic.

To create an access list, specify the protocol to be filtered, assign a unique name or number to the access list, and define packet filtering criteria. A single access list can have multiple filtering statements.

We recommend that you create access lists on a TFTP server and then download these access lists to the required device to simplify the maintenance of access lists. For details, see the "Create or Edit Access List Statements on a TFTP Server" section.

Assign a Unique Name or Number to Each Access Control List

When configuring access lists on a device, you must identify each access list uniquely within a protocol by assigning either a name or a number to that protocol's access list. Access lists of some protocols must be identified by a name, and access lists of other protocols must be identified by a number. Some protocols can be identified by either a name or a number. When a number is used to identify an access list, the number must be within the specific range of numbers that is valid for the protocol.

You can specify access lists by names for the following protocols:

- Apollo Domain
- Internetwork Packet Exchange (IPX)
- IP
- ISO Connectionless Network Service (CLNS)
- NetBIOS IPX
- Source-route bridging NetBIOS

You can specify access lists by numbers for the protocols listed in the table below.

Table 3: Protocols with Access Lists Specified by Numbers

Protocol	Range
AppleTalk	300–399

Protocol	Range
DECnet and extended DECnet	600–699
Ethernet address	700–799
Ethernet type code	200–299
Extended IP	100–199, 2000–2699
Extended IPX	900–999
Extended transparent bridging	1100–1199
Extended Virtual Integrated Network Service (VINES)	101–200
Extended Xerox Network Systems (XNS)	500–599
IP	1–99, 1300–1999
IPX	800–899
IPX Service Advertising Protocol (SAP)	1000–1099
Simple VINES	201–300
Source-route bridging (protocol type)	200–299
Source-route bridging (vendor code)	700–799
Standard VINES	1–100
Transparent bridging (protocol type)	200–299
Transparent bridging (vendor code)	700–799
XNS	400–499

Define Criteria for Forwarding or Blocking Packets

When creating an access list, define criteria that are applied to each packet that is processed by the device so that the device can forward or block each packet based on whether or not the packet matches the criteria.

Typical criteria that you define in access lists include packet source addresses, packet destination addresses, and upper-layer protocol of the packet. However, each protocol has its own specific set of criteria that can be defined.

In a single access list, you can define multiple criteria in separate access list statements. Each of these statements must reference the same identifying name or number to bind statements to the same access list. You can have as many criteria statements as you want, limited only by the available memory of the device. The more statements there are in an access list, the more difficult it will be to comprehend and manage an access list.

Deny All Traffic Criteria Statement

At the end of every access list is an implied "deny all traffic" criteria statement. This statement implies that if a packet does not match any criteria statement, the packet will be blocked.

Note For most protocols, if you define an inbound access list for traffic filtering, you should include explicit access list criteria statements to permit routing updates. If you do not, you might effectively lose communication from the interface when routing updates are blocked by the "deny all traffic" statement at the end of the access list.

Order of Criteria Statements

Each criteria statement that you enter is appended to the end of the access list statements. You cannot delete individual statements after they are created. You can delete only an entire access list.

The order of access list statements in an access list is important. When a device is deciding whether to forward or block a packet, Cisco software tests the packet against each criteria statement in the order in which the statements were created. After a match is found, no more criteria statements are checked.

If you create a criteria statement that explicitly permits all traffic, statements added later will not be checked. If you need additional statements, you must delete the access list and configure a new access list.

Create or Edit Access Control List Statements on a TFTP Server

Because the order of access list criteria statements is important and you cannot reorder or delete criteria statements on your device, we recommend that you create all access list statements on a TFTP server and that you download the entire access list to your device.

Create access list statements using any text editor, and save access list statements in ASCII format to a TFTP server that is accessible from your device. Then, on your device, use the **copy tftp:** *file-id* **system:running-config** command to copy the access list from the TFTP server to your device. Finally, use the **copy system:running-config nvram:startup-config** command to save the access list to your device's NVRAM.

If you want to make changes to an access list, you can make them to the text file on the TFTP server and copy the edited file to your device.

Note The first command of an edited access list file should delete the previous access list (for example, use the **no access-list** command at the beginning of the file). If you do not delete the previous version of the access list, when you copy the edited file to your device you will merely be appending additional criteria statements to the end of the existing access list.

Apply an Access Control List to an Interface

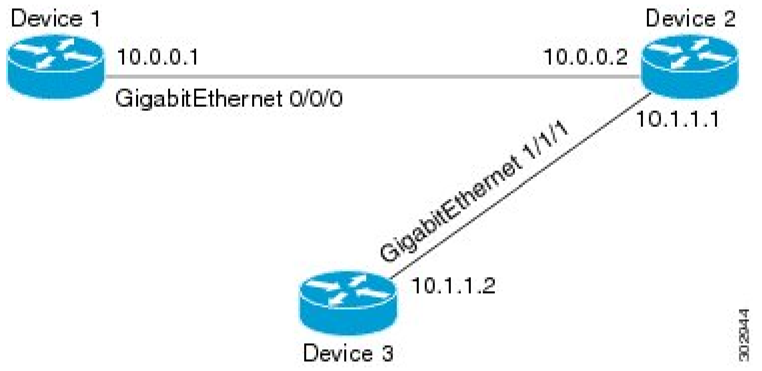
With some protocols, you can apply up to two access lists to an interface: one inbound access list and one outbound access list. With other protocols, you apply only one access list that checks both inbound and outbound packets.

If the access list is inbound, when a device receives a packet, Cisco software checks the access list's criteria statements for a match. If the packet is permitted, the software continues to process the packet. If the packet is denied, the software discards the packet.

If the access list is outbound, after receiving and routing a packet to the outbound interface, Cisco software checks the access list's criteria statements for a match. If the packet is permitted, the software transmits the packet. If the packet is denied, the software discards the packet.

Note Access lists that are applied to interfaces on a device do not filter traffic that originates from that device.

Figure 2: Topology for Applying Access Control Lists



The figure above shows that Device 2 is a bypass device that is connected to Device 1 and Device 3. An outbound access list is applied to Gigabit Ethernet interface 0/0/0 on Device 1. When you ping Device 3 from Device 1, the access list does not check for packets going outbound because the traffic is locally generated.

The access list check is bypassed for locally generated packets, which are always outbound.

By default, an access list that is applied to an outbound interface for matching locally generated traffic will bypass the outbound access list check; but transit traffic is subjected to the outbound access list check.

Note The behavior described above applies to all single-CPU platforms that run Cisco software.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases

Related Topic	Document Title
IP access list commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
Dynamic access lists	"Configuring Lock-and-Key Security (Dynamic Access Lists)"
Reflexive access lists	"Configuring IP Session Filtering (Reflexive Access Lists)"

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

CHAPTER 3

IPv6 Access Control Lists

Access lists determine what traffic is blocked and what traffic is forwarded at device interfaces and allow filtering of traffic based on source and destination addresses, and inbound and outbound traffic to a specific interface. Standard IPv6 ACL functionality was extended to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control. Standard IPv6 ACL functionality was extended to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control.

This module describes how to configure IPv6 traffic filtering and to control access to virtual terminal lines.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About IPv6 Access Control Lists

Access Control Lists for IPv6 Traffic Filtering

The standard ACL functionality in IPv6 is similar to standard ACLs in IPv4. Access lists determine what traffic is blocked and what traffic is forwarded at device interfaces and allow filtering based on source and destination addresses, inbound and outbound to a specific interface. Each access list has an implicit deny statement at the end. IPv6 ACLs are defined and their deny and permit conditions are set using the **ipv6 access-list**command with the **deny** and **permit** keywords in global configuration mode.

IPv6 extended ACLs augments standard IPv6 ACL functionality to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control (functionality similar to extended ACLs in IPv4).

How to Configure IPv6 Access Control Lists

Configuring IPv6 Traffic Filtering

Creating and Configuring an IPv6 ACL for Traffic Filtering

This section describes how to configure your networking devices to filter traffic, function as a firewall, or detect potential viruses.

Before You Begin

Note

- Each IPv6 ACL contains implicit permit rules to enable IPv6 neighbor discovery. These rules can be overridden by the user by placing a deny ipv6 any any statement within an ACL. The IPv6 neighbor discovery process makes use of the IPv6 network layer service; therefore, by default, IPv6 ACLs implicitly allow IPv6 neighbor discovery packets to be sent and received on an interface. In IPv4, the Address Resolution Protocol (ARP), which is equivalent to the IPv6 neighbor discovery process, makes use of a separate data link layer protocol; therefore, by default, IPv4 ACLs implicitly allow ARP packets to be sent and received on an interface.
- Time-based and reflexive ACLs are not supported for IPv4 or IPv6 on the Cisco 12000 series platform. The **reflect**, **timeout**, and **time-range** keywords of the **permit** command in IPv6 are excluded on the Cisco 12000 series.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. Do one of the following:
   - **permit protocol** {*source-ipv6-prefix* / *prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix* / *prefix-length*| **any** | **host** *destination-ipv6-address*| **auth**} [*operator* [*port-number*]] [**dest-option-type** [*doh-number*| *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**reflect** *name* [**timeout** *value*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*]
   -
   -
   - **deny** *protocol* {*source-ipv6-prefix* / *prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* *port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 access-list** *access-list-name* **Example:** `Router(config)# ipv6 access-list outbound`	Defines an IPv6 ACL, and enters IPv6 access list configuration mode. • The *access-list name* argument specifies the name of the IPv6 ACL. IPv6 ACL names cannot contain a space or quotation mark, or begin with a numeral.
Step 4	Do one of the following: • **permit protocol** {*source-ipv6-prefix* / *prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]]	Specifies permit or deny conditions for an IPv6 ACL.

	Command or Action	Purpose

	{*destination-ipv6-prefix* / *prefix-length*| **any** | **host** *destination-ipv6-address*| **auth**} [*operator* [*port-number*]] [**dest-option-type** [*doh-number*| *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**reflect** *name* [**timeout** *value*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*]
	•
	•
	• **deny** *protocol* {*source-ipv6-prefix* / *prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**]
	
	Example:
	`Router(config-ipv6-acl)# permit tcp 2001:DB8:0300:0201::/32 eq telnet any reflect reflectout`
	
	Example:
	
	Example:
	
	Example:
	`Router(config-ipv6-acl)# deny tcp host 2001:DB8:1::1 any log-input`	

Applying the IPv6 ACL to an Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 traffic-filter** *access-list-name* {**in**\| **out**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface ethernet 0`	Specifies the interface type and number, and enters interface configuration mode.

Step 4	**ipv6 traffic-filter** *access-list-name* {**in**| **out**}
	
	Example:
	
	`Router(config-if)# ipv6 traffic-filter outbound out`
		Applies the specified IPv6 access list to the interface specified in the previous step.

Controlling Access to a vty

Creating an IPv6 ACL to Provide Access Class Filtering

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. Do one of the following:
   - **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*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*
   - **deny** *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*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 access-list** *access-list-name* **Example:** `Device(config)# ipv6 access-list cisco`	Defines an IPv6 ACL, and enters IPv6 access list configuration mode.

	Command or Action	Purpose

Step 4	Do one of the following:
	• **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*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*
	• **deny** *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*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**
	
	Example:
	`Device(config-ipv6-acl)# permit ipv6 host 2001:DB8:0:4::32 any`
	
	Example:
	`Device(config-ipv6-acl)# deny ipv6 host 2001:DB8:0:6::6 any`
		Specifies permit or deny conditions for an IPv6 ACL.

Applying an IPv6 ACL to the Virtual Terminal Line

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **line** [**aux**\| **console**\| **tty**\| **vty**] *line-number*[*ending-line-number*]
4. **ipv6 access-class** *ipv6-access-list-name* {**in**\| **out**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**line** [**aux**| **console**| **tty**| **vty**] *line-number*[*ending-line-number*] **Example:** `Device(config)# line vty 0 4`	Identifies a specific line for configuration and enters line configuration mode. • In this example, the **vty** keyword is used to specify the virtual terminal lines for remote console access.
Step 4	**ipv6 access-class** *ipv6-access-list-name* {**in**| **out**} **Example:** `Device(config-line)# ipv6 access-class cisco in`	Filters incoming and outgoing connections to and from the device based on an IPv6 ACL.

Configuration Examples for IPv6 Access Control Lists

Example: Verifying IPv6 ACL Configuration

In this example, the **show ipv6 access-list** command is used to verify that IPv6 ACLs are configured correctly:

```
Device> show ipv6 access-list

IPv6 access list inbound
    permit tcp any any eq bgp (8 matches) sequence 10
    permit tcp any any eq telnet  (15 matches) sequence 20
    permit udp any any  sequence 30

IPv6 access list Virtual-Access2.1#427819008151 (per-user)
    permit tcp host 2001:DB8:1::32 eq bgp host 2001:DB8:2::32 eq 11000 sequence 1
    permit tcp host 2001:DB8:1::32 eq telnet host 2001:DB8:2::32 eq 11001 sequence 2
```

Example: Creating and Applying an IPv6 ACL

The following example shows how to restrict HTTP access to certain hours during the day and log any activity outside of the permitted hours:

```
Device# configure terminal
Device(config)# time-range lunchtime
Device(config-time-range)# periodic weekdays 12:00 to 13:00
Device(config-time-range)# exit
Device(config)# ipv6 access-list INBOUND
Device(config-ipv6-acl)# permit tcp any any eq www time-range lunchtime
Device(config-ipv6-acl)# deny tcp any any eq www log-input
Device(config-ipv6-acl)# permit tcp 2001:DB8::/32 any
Device(config-ipv6-acl)# permit udp 2001:DB8::/32 any
Device(config-ipv6-acl)# end
```

Example: Controlling Access to a vty

In the following example, incoming connections to the virtual terminal lines 0 to 4 are filtered based on the IPv6 access list named acl1:

```
ipv6 access-list acl1
 permit ipv6 host 2001:DB8:0:4::2/32 any
!
line vty 0 4
 ipv6 access-class acl1 in
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	*Cisco IOS IPv6 Command Reference*
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
CISCO-UNIFIED-FIREWALL-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for IPv6 Access Control Lists

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 4: Feature Information for IPv6 Access Control Lists

Feature Name	Releases	Feature Information
IPv6 Services: Standard Access Control Lists	12.0(22)S 12.2(14)S 12.2(28)SB 12.2(25)SG 12.2(33)SRA 12.2(17a)SX1 12.2(2)T 12.3 12.3(2)T 12.4 12.4(2)T 15.0(1)S	Access lists determine what traffic is blocked and what traffic is forwarded at router interfaces and allow filtering based on source and destination addresses, inbound and outbound to a specific interface.
IPv6 Services: Extended Access Control Lists	12.0(23)S 12.2(14)S 12.2(28)SB 12.2(25)SG 12.2(33)SRA 12.2(17a)SX1 12.2(13)T 12.3 12.3(2)T 12.4 12.4(2)T 15.0(1)S 15.4(3)S	Standard IPv6 ACL functionality was extended to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control.

CHAPTER 4

IPv6 ACL Extensions for IPsec Authentication Headers

The IPv6 ACL Extensions for IPsec Authentication Headers feature allows TCP or UDP parsing when an IPv6 IPsec authentication header is present.

This module describes how to configure TCP or UDP matching regardless of whether an authentication header (AH) is present or absent.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About IPv6 ACL Extensions for IPsec Authentication Header

IPv6 ACL Extensions for IPsec Authentication Header

This feature provides the ability to match on the upper layer protocol (ULP) (for example, TCP, User Datagram Protocol [UDP], ICMP, SCTP) regardless of whether an authentication header (AH) is present or absent.

TCP or UDP traffic can be matched to the upper-layer protocol (ULP) (for example, TCP, UDP, ICMP, SCTP) if an AH is present or absent. Before this feature was introduced, this function was only available if an AH was absent.

This feature introduces the keyword **auth** to the **permit**and **deny**commands. The **auth** keyword allows matching traffic against the presence of the authentication header in combination with the specified protocol; that is, TCP or UDP.

IPv6 traffic can be matched to a ULP when an AH header is present. To perform this function, enter the **ahp** option for the *protocol* argument when using the **permit** or **deny** command.

How to Configure IPv6 ACL Extensions for IPsec Authentication Header

Configuring TCP or UDP Matching

TCP or UDP traffic can be matched to the ULP (for example, TCP, UDP, ICMP, SCTP) if an AH is present or absent. Before this feature was introduced, this function was only available if an AH was absent.

Use of the keyword **auth** with the **permit icmp** and **deny icmp** commands allows TCP or UDP traffic to be matched to the ULP if an AH is present. TCP or UDP traffic without an AH will not be matched.

IPv6 traffic can be matched to a ULP when an AH header is present. To perform this function, enter the **ahp** option for the *protocol* argument when using the **permit** or **deny** command.

Perform this task to allow TCP or UDP traffic to be matched to the ULP if an AH is present.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. **permit icmp auth**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router# enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 access-list** *access-list-name* **Example:** `Router(config)# ipv6 access-list list1`	Defines an IPv6 access list and places the router in IPv6 access list configuration mode.

Step 4	**permit icmp auth**
	
	Example:
	
	Example:
	
	`or`
	
	Example:
	
	deny icmp auth
	
	Example:
	
	`Router(config-ipv6-acl)# permit icmp auth`
		Specifies permit or deny conditions for an IPv6 ACL using the **auth** keyword, which is used to match against the presence of the AH.

Configuration Examples for IPv6 ACL Extensions for IPsec Authentication Header

Example: Configuring TCP or UDP Matching

The following example allows any TCP traffic regardless of whether or not an AH is present:

```
IPv6 access list example1
    permit tcp any any
```

The following example allows TCP or UDP parsing only when an AH header is present. TCP or UDP traffic without an AH will not be matched:

```
IPv6 access list example2
    deny tcp host 2001::1 any log sequence 5
    permit tcp any any auth sequence 10
    permit udp any any auth sequence 20
```

The following example allows any IPv6 traffic containing an authentication header:

```
IPv6 access list example3
    permit ahp any any
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	*Cisco IOS IPv6 Command Reference*
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for IPv6 ACL Extensions for IPsec Authentication Header

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5: Feature Information for IPv6 ACL Extensions for IPsec Authentication Header

Feature Name	Releases	Feature Information
IPv6 ACL Extensions for IPsec Authentication Header	12.4(20)T	The IPv6 ACL extensions for IPsec authentication headers feature allows TCP or UDP parsing when an IPv6 IPsec authentication header is present. The following commands were introduced or modified: **deny**, **ipv6 access-list**, **permit**.

CHAPTER 5

IPv6 ACL Extensions for Hop by Hop Filtering

The IPv6 ACL Extensions for Hop by Hop Filtering feature allows you to control IPv6 traffic that might contain hop-by-hop extension headers. You can configure an access control list (ACL) to deny all hop-by-hop traffic or to selectively permit traffic based on protocol.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About IPv6 ACL Extensions for Hop by Hop Filtering

ACLs and Traffic Forwarding

IPv6 access control lists (ACLs) determine what traffic is blocked and what traffic is forwarded at device interfaces. ACLs allow filtering based on source and destination addresses, inbound and outbound to a specific interface. Use the **ipv6 access-list** command to define an IPv6 ACL, and the **deny** and **permit** commands to configure its conditions.

The IPv6 ACL Extensions for Hop by Hop Filtering feature implements RFC 2460 to support traffic filtering in any upper-layer protocol type.

How to Configure IPv6 ACL Extensions for Hop by Hop Filtering

Configuring IPv6 ACL Extensions for Hop by Hop Filtering

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. **permit** *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* | *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**reflect** *name* [**timeout** *value*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*]
5. **deny** *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* | *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**]
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 access-list** *access-list-name* **Example:** `Device(config)# ipv6 access-list hbh-acl`	Defines an IPv6 ACL and enters IPv6 access list configuration mode.
Step 4	**permit** *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* |	Sets permit conditions for the IPv6 ACL.

	Command or Action	Purpose
	auth} [*operator* [*port-number*]] [**dest-option-type** [*header-number* | *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**reflect** *name* [**timeout** *value*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] **Example:** `Device(config-ipv6-acl)# permit icmp any any dest-option-type`	
Step 5	**deny** *protocol* {*source-ipv6-prefix*/*prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix*/*prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* | *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**] **Example:** `Device(config-ipv6-acl)# deny icmp any any dest-option-type`	Sets deny conditions for the IPv6 ACL.

Step 6	**end**
	
	Example:
	`Device (config-ipv6-acl)# end`
		Returns to privileged EXEC configuration mode.

Configuration Example for IPv6 ACL Extensions for Hop by Hop Filtering

Example: IPv6 ACL Extensions for Hop by Hop Filtering

```
Device(config)# ipv6 access-list hbh_acl
Device(config-ipv6-acl)# permit tcp any any hbh
Device(config-ipv6-acl)# permit tcp any any
Device(config-ipv6-acl)# permit udp any any
Device(config-ipv6-acl)# permit udp any any hbh
Device(config-ipv6-acl)# permit hbh any any
Device(config-ipv6-acl)# permit any any
Device(config-ipv6-acl)# hardware statistics
Device(config-ipv6-acl)# exit

! Assign an IP address and add the ACL on the interface.

Device(config)# interface FastEthernet3/1
Device(config-if)# ipv6 address 1001::1/64
Device(config-if)# ipv6 traffic-filter hbh_acl in
Device(config-if)# exit
Device(config)# exit
Device# clear counters
Clear "show interface" counters on all interfaces [confirm]
Device#

! Verify the configurations.

Device# show running-config interface FastEthernet3/1

Building configuration...

Current configuration : 114 bytes
!
interface FastEthernet3/1
no switchport
ipv6 address 1001::1/64
ipv6 traffic-filter hbh_acl
end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Security commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
IPv6 addressing and basic connectivity	*IPv6 Addressing and Basic Connectivity Configuration Guide*
IPv6 features	IPv6 Feature Mapping
RFCs for IPv6	*IPv6 RFCs*

Standards and RFCs

Standard/RFC	Title
RFC 2460	*Internet Protocol, Version 6 (IPv6)*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for IPv6 ACL Extensions for Hop by Hop Filtering

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 6: Feature Information for IPv6 ACL Extensions for Hop by Hop Filtering

Feature Name	Releases	Feature Information
IPv6 ACL Extensions for Hop by Hop Filtering	15.1(1)SG 15.1(1)SY 15.2(3)T 15.3(1)S	Allows you to control IPv6 traffic that might contain hop-by-hop extension headers. The following commands were introduced or modified: **deny** (IPv6), **permit** (IPv6).

CHAPTER 6

Creating an IP Access List and Applying It to an Interface

IP access lists provide many benefits for securing a network and achieving nonsecurity goals, such as determining quality of service (QoS) factors or limiting **debug** command output. This module describes how to create standard, extended, named, and numbered IP access lists. An access list can be referenced by a name or a number. Standard access lists filter on only the source address in IP packets. Extended access lists can filter on source address, destination address, and other fields in an IP packet.

After you create an access list, you must apply it to something in order for it to have any effect. This module describes how to apply an access list to an interface. However, there are many other uses for an access list, which are referenced in this module and described in other modules and in other configuration guides for various technologies.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Creating an IP Access List and Applying It to an Interface

Before you create or apply an IP access list, you should understand the concepts in the “IP Access List Overview” module. You should also have IP running in your network.

Information About Creating an IP Access List and Applying It to an Interface

Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list**command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.
- While you are creating an access list or after it is created, you might want to delete an entry.
    - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.

   - You can delete an entry from a named access list. Use the **no permit**or **no deny** command to delete the appropriate entry.

- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.
- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

Access List Remarks

You can include comments or remarks about entries in any IP access list. An access list remark is an optional remark before or after an access list entry that describes the entry so that you do not have to interpret the purpose of the entry. Each remark is limited to 100 characters in length.

The remark can go before or after a **permit** or **deny** statement. Be consistent about where you add remarks. Users may be confused if some remarks precede the associated **permit** or **deny** statements and some remarks follow the associated statements.

The following is an example of a remark that describes function of the subsequent **deny** statement:

```
ip access-list extended telnetting
 remark Do not allow host1 subnet to telnet out
 deny tcp host 172.16.2.88 any eq telnet
```

Additional IP Access List Features

Beyond the basic steps to create a standard or extended access list, you can enhance your access lists as mentioned below. Each of these methods is described completely in the *Refining an IP Access List module*.

- You can impose dates and times when **permit** or **deny** statements in an extended access list are in effect, making your access list more granular and specific to an absolute or periodic time period.
- After you create a named or numbered access list, you might want to add entries or change the order of the entries, known as resequencing an access list.
- You can achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

How to Create an IP Access List and Apply It to an Interface

This section describes the general ways to create a standard or extended access list using either a name or a number. Access lists are very flexible; the tasks simply illustrate one **permit** command and one **deny** command to provide you the command syntax of each. Only you can determine how many **permit** and **deny** commands you need and their order.

Note The first two tasks in this module create an access list; you must apply the access list in order for it to function. If you want to apply the access list to an interface, perform the task "Applying the Access List to an Interface". If you don’t intend to apply the access list to an interface, see the "Where to Go Next" for pointers to modules that describe other ways to apply access lists.

Creating a Standard Access List to Filter on Source Address

If you want to filter on source address only, a standard access list is simple and sufficient. There are two alternative types of standard access list: named and numbered. Named access lists allow you to identify your access lists with a more intuitive name rather than a number, and they also support more features than numbered access lists.

Creating a Named Access List to Filter on Source Address

Use a standard, named access list if you need to filter on source address only. This task illustrates one **permit** statement and one **deny** statement, but the actual statements you use and their order depend on what you want to filter or allow. Define your **permit** and **deny** statements in the order that achieves your filtering goals.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list standard** *name*
4. **remark** *remark*
5. **deny** {*source* [*source-wildcard*] | **any**} [**log**]
6. **remark** *remark*
7. **permit** {*source* [*source-wildcard*] | **any**} [**log**]
8. Repeat some combination of Steps 4 through 7 until you have specified the sources on which you want to base your access list.
9. **end**
10. **show ip access-list**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list standard** *name* **Example:** `Device(config)# ip access-list standard R&D`	Defines a standard IP access list using a name and enters standard named access list configuration mode.
Step 4	**remark** *remark* **Example:** `Device(config-std-nacl)# remark deny Sales network`	(Optional) Adds a user-friendly comment about an access list entry. • A remark can precede or follow an access list entry. • In this example, the remark reminds the network administrator that the subsequent entry denies the Sales network access to the interface (assuming this access list is later applied to an interface).
Step 5	**deny** {*source* [*source-wildcard*] | **any**} [**log**] **Example:** `Device(config-std-nacl)# deny 172.16.0.0 0.0.255.255 log`	(Optional) Denies the specified source based on a source address and wildcard mask. • If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address. • Optionally use the keyword **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, all hosts on network 172.16.0.0 are denied passing the access list. • Because this example explicitly denies a source address and the **log** keyword is specified, any packets from that source are logged when they are denied. This is a way to be notified that someone on a network or host is trying to gain access.
Step 6	**remark** *remark* **Example:** `Device(config-std-nacl)# remark Give access to Tester's host`	(Optional) Adds a user-friendly comment about an access list entry. • A remark can precede or follow an access list entry. • This remark reminds the network administrator that the subsequent entry allows the Tester's host access to the interface.
Step 7	**permit** {*source* [*source-wildcard*] | **any**} [**log**] **Example:** `Device(config-std-nacl)# permit 172.18.5.22 0.0.0.0`	Permits the specified source based on a source address and wildcard mask. • Every access list needs at least one **permit** statement; it need not be the first entry. • If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.

	Command or Action	Purpose
		• Optionally use the keyword **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, host 172.18.5.22 is allowed to pass the access list.
Step 8	Repeat some combination of Steps 4 through 7 until you have specified the sources on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 9	**end**
	Example:
	`Device(config-std-nacl)# end`
		Exits standard named access list configuration mode and enters privileged EXEC mode.

Step 10	**show ip access-list** **Example:** `Device# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

What to Do Next

The access list you created is not in effect until you apply it to an interface, a vty line, or reference it from a command that uses an access list. See "Applying the Access List to an Interface" or "Where to Go Next" for pointers to modules that describe other ways to use access lists.

Creating a Numbered Access List to Filter on Source Address

Configure a standard, numbered access list if you need to filter on source address only and you prefer not to use a named access list.

IP standard access lists are numbered 1 to 99 or 1300 to 1999. This task illustrates one **permit** statement and one **deny** statement, but the actual statements you use and their order depend on what you want to filter or allow. Define your **permit** and **deny** statements in the order that achieves your filtering goals.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number permit* {*source* [*source-wildcard*] | **any**} [**log**]
4. **access-list** *access-list-number* **deny** {*source* [*source-wildcard*] | **any**} [**log**]
5. Repeat some combination of Steps 3 through 6 until you have specified the sources on which you want to base your access list.
6. **end**
7. **show ip access-list**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**access-list** *access-list-number permit* {*source* [*source-wildcard*] | **any**} [**log**] **Example:** `Device(config)# access-list 1 permit 172.16.5.22 0.0.0.0`	Permits the specified source based on a source address and wildcard mask. • Every access list needs at least one permit statement; it need not be the first entry. • Standard IP access lists are numbered 1 to 99 or 1300 to 1999. • If the source-wildcard is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address. • Optionally use the keyword any as a substitute for the source source-wildcard to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, host 172.16.5.22 is allowed to pass the access list.
Step 4	**access-list** *access-list-number* **deny** {*source* [*source-wildcard*] | **any**} [**log**] **Example:** `Device(config)# access-list 1 deny 172.16.7.34 0.0.0.0`	Denies the specified source based on a source address and wildcard mask. • If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.

	Command or Action	Purpose
		• Optionally use the abbreviation **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, host 172.16.7.34 is denied passing the access list.
Step 5	Repeat some combination of Steps 3 through 6 until you have specified the sources on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 6	**end**
	
	Example:
	
	`Device(config)# end`
		Exits global configuration mode and enters privileged EXEC mode.

Step 7	**show ip access-list** **Example:** `Device# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

What to Do Next

The access list you created is not in effect until you apply it to an interface, a vty line, or reference it from a command that uses an access list. See "Applying the Access List to an Interface" or "Where to Go Next" for pointers to modules that describe other ways to use access lists.

Creating an Extended Access List

If you want to filter on anything other than source address, you need to create an extended access list. There are two alternative types of extended access list: named and numbered. Named access lists allow you to identify your access lists with a more intuitive name rather than a number, and they also support more features.

For details on how to filter something other than source or destination address, see the syntax descriptions in the command reference documentation.

Creating a Named Extended Access List

Create a named extended access list if you want to filter on source and destination address, or a combination of addresses and other IP fields.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *name*
4. **remark** *remark*
5. **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
6. **remark** *remark*
7. **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
8. Repeat some combination of Steps 4 through 7 until you have specified the fields and values on which you want to base your access list.
9. **end**
10. **show ip access-list**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *name* **Example:** `Router(config)# ip access-list extended nomarketing`	Defines an extended IP access list using a name and enters extended named access list configuration mode.
Step 4	**remark** *remark* **Example:** `Router(config-ext-nacl)# remark protect server by denying access from the Marketing network`	(Optional) Adds a user-friendly comment about an access list entry. • A remark can precede or follow an access list entry. • In this example, the remark reminds the network administrator that the subsequent entry denies the Sales network access to the interface.

	Command or Action	Purpose
Step 5	**deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config-ext-nacl)# deny ip 172.18.0.0 0.0.255.255 host 172.16.40.10 log`	(Optional) Denies any packet that matches all of the conditions specified in the statement. • If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255. • Optionally use the keyword **host** *source* to indicate a source and source wildcard of *source* 0.0.0.0 or the abbreviation **host** *destination*to indicate a destination and destination wildcard of *destination* 0.0.0.0. • In this example, packets from the source network 172.18.0.0 are denied access to host 172.16.40.10. Logging messages about packets permitted or denied by the access list are sent to the facility configured by the **logging facility** command (for example, console, terminal, or syslog). That is, any packet that matches the access list will cause an informational logging message about the packet to be sent to the configured facility. The level of messages logged to the console is controlled by the **logging console** command.
Step 6	**remark** *remark* **Example:** `Router(config-ext-nacl)# remark allow TCP from any source to any destination`	(Optional) Adds a user-friendly comment about an access list entry. • A remark can precede or follow an access list entry.
Step 7	**permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config-ext-nacl)# permit tcp any any`	Permits any packet that matches all of the conditions specified in the statement. • Every access list needs at least one permit statement. • If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255. • In this example, TCP packets are allowed from any source to any destination. • The **log-input** keyword can be configured, but it is not supported, and will not work as expected.

	Command or Action	Purpose
Step 8	Repeat some combination of Steps 4 through 7 until you have specified the fields and values on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 9	**end**
	
	Example:
	
	`Router(config-ext-nacl)# end`
		Ends configuration mode and brings the system to privileged EXEC mode.

Step 10	**show ip access-list** **Example:** `Router# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

What to Do Next

The access list you created is not in effect until you apply it to an interface, a vty line, or reference it from a command that uses an access list. See "Applying the Access List to an Interface" or the "Where to Go Next" for pointers to modules that describe other ways to use access lists.

Creating a Numbered Extended Access List

Create a numbered extended access list if you want to filter on source and destination address, or a combination of addresses and other IP fields, and you prefer not to use a name. Extended IP access lists are numbered 100 to 199 or 2000 to 2699.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* **remark** *remark*
4. **access-list** *access-list-number* **permit** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
5. **access-list** *access-list-number* **remark** *remark*
6. **access-list** *access-list-number* **deny** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
7. Repeat some combination of Steps 3 through 6 until you have specified the fields and values on which you want to base your access list.
8. **end**
9. **show ip access-list**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**access-list** *access-list-number* **remark** *remark* **Example:** `Router(config)# access-list 107 remark allow Telnet packets from any source to network 172.69.0.0 (headquarters)`	(Optional) Adds a user-friendly comment about an access list entry. • A remark of up to 100 characters can precede or follow an access list entry.
Step 4	**access-list** *access-list-number* **permit** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config)# access-list 107 permit tcp any 172.69.0.0 0.0.255.255 eq telnet`	Permits any packet that matches all of the conditions specified in the statement. • Every access list needs at least one **permit** statement; it need not be the first entry. • Extended IP access lists are numbered 100 to 199 or 2000 to 2699. • If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255. • TCP and other protocols have additional syntax available. See the **access-list** command in the command reference for complete syntax.
Step 5	**access-list** *access-list-number* **remark** *remark* **Example:** `Router(config)# access-list 107 remark deny all other TCP packets`	(Optional) Adds a user-friendly comment about an access list entry. • A remark of up to 100 characters can precede or follow an access list entry.
Step 6	**access-list** *access-list-number* **deny** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence**	Denies any packet that matches all of the conditions specified in the statement.

	Command or Action	Purpose
	precedence] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config)# access-list 107 deny tcp any any`	• If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.
Step 7	Repeat some combination of Steps 3 through 6 until you have specified the fields and values on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 8	**end**
	
	Example:
	
	`Router(config)# end`
		Ends configuration mode and brings the system to privileged EXEC mode.

Step 9	**show ip access-list** **Example:** `Router# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

Applying the Access List to an Interface

Perform this task to apply an access list to an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip access-group** {*access-list-number* | *access-list-name*} {**in** | **out**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface ethernet 0`	Specifies an interface and enters interface configuration mode.

Step 4	**ip access-group** {*access-list-number* | *access-list-name*} {**in** | **out**}
	
	Example:
	
	`Router(config-if)# ip access-group noncorp in`
		Applies the specified access list to the incoming or outgoing interface.
		• When you are filtering on source addresses, you typically apply the access list to an incoming interface.
		• Filtering on source addresses is most efficient when applied near the destination.

What to Do Next

The access list you created is not in effect until you apply it to an interface, a vty line, or reference it from a command that uses an access list. See "Applying the Access List to an Interface" or "Where to Go Next" for pointers to modules that describe other ways to use access lists.

Configuration Examples for Creating an IP Access List and Applying It to an Interface

Example Filtering on Source Address (Hosts)

In the following example, the workstation belonging to Jones is allowed access to Ethernet interface 0 and the workstation belonging to Smith is not allowed access:

```
interface ethernet 0
 ip access-group workstations in
!
ip access-list standard workstations
 remark Permit only Jones workstation through
 permit 172.16.2.88
 remark Do not allow Smith workstation through
 deny 172.16.3.13
```

Example Filtering on Source Address (Subnet)

In the following example, the Jones subnet is not allowed access to Ethernet interface 0, but the Main subnet is allowed access:

```
interface ethernet 0
 ip access-group prevention in
!
ip access-list standard prevention
 remark Do not allow Jones subnet through
 deny 172.22.0.0 0.0.255.255
 remark Allow Main subnet
 permit 172.25.0.0 0.0.255.255
```

Example Filtering on Source Address Destination Address and IP Protocols

The following configuration example shows an interface with two access lists, one applied to outgoing packets and one applied to incoming packets. The standard access list named Internet_filter filters outgoing packets on source address. The only packets allowed out the interface must be from source 172.16.3.4.

The extended access list named marketing_group filters incoming packets. The access list permits Telnet packets from any source to network 172.26.0.0 and denies all other TCP packets. It permits any ICMP packets. It denies UDP packets from any source to network 172.26.0 0 on port numbers less than 1024. Finally, the access list denies all other IP packets and performs logging of packets passed or denied by that entry.

```
interface Ethernet0/5
 ip address 172.20.5.1 255.255.255.0
 ip access-group Internet_filter out
 ip access-group marketing_group in
!
ip access-list standard Internet_filter
 permit 172.16.3.4
ip access-list extended marketing_group
 permit tcp any 172.26.0.0 0.0.255.255 eq telnet
 deny tcp any any
 permit icmp any any
 deny udp any 172.26.0.0 0.0.255.255 lt 1024
 deny ip any any
```

Example Filtering on Source Address (Host and Subnets) Using a Numbered Access List

In the following example, network 10.0.0.0 is a Class A network whose second octet specifies a subnet; that is, its subnet mask is 255.255.0.0. The third and fourth octets of a network 10.0.0.0 address specify a particular host. Using access list 2, the Cisco IOS software would accept one address on subnet 48 and reject all others on that subnet. The last line of the list shows that the software would accept addresses on all other network 10.0.0.0 subnets.

```
interface ethernet 0
 ip access-group 2 in
!
access-list 2 permit 10.48.0.3
access-list 2 deny 10.48.0.0  0.0.255.255
access-list 2 permit 10.0.0.0  0.255.255.255
```

Example Preventing Telnet Access to a Subnet

In the following example, the Jones subnet is not allowed to Telnet out Ethernet interface 0:

```
interface ethernet 0
 ip access-group telnetting out
!
ip access-list extended telnetting
 remark Do not allow Jones subnet to telnet out
 deny tcp 172.20.0.0 0.0.255.255 any eq telnet
 remark Allow Top subnet to telnet out
 permit tcp 172.33.0.0 0.0.255.255 any eq telnet
```

Example Filtering on TCP and ICMP Using Port Numbers

In the following example, the first line of the extended access list named goodports permits any incoming TCP connections with destination ports greater than 1023. The second line permits incoming TCP connections to the Simple Mail Transfer Protocol (SMTP) port of host 172.28.1.2. The last line permits incoming ICMP messages for error feedback.

```
interface ethernet 0
 ip access-group goodports in
!
ip access-list extended goodports
 permit tcp any 172.28.0.0 0.0.255.255 gt 1023
 permit tcp any host 172.28.1.2 eq 25
 permit icmp any 172.28.0.0 255.255.255.255
```

Example Allowing SMTP (E-mail) and Established TCP Connections

Suppose you have a network connected to the Internet, and you want any host on an Ethernet to be able to form TCP connections to any host on the Internet. However, you do not want IP hosts to be able to form TCP connections to hosts on the Ethernet except to the mail (SMTP) port of a dedicated mail host.

SMTP uses TCP port 25 on one end of the connection and a random port number on the other end. The same two port numbers are used throughout the life of the connection. Mail packets coming in from the Internet will have a destination port of 25. Outbound packets will have the port numbers reversed. The fact that the secure system behind the router always will accept mail connections on port 25 is what makes possible separate control of incoming and outgoing services. The access list can be configured on either the outbound or inbound interface.

In the following example, the Ethernet network is a Class B network with the address 172.18.0.0, and the address of the mail host is 172.18.1.2. The **established**keyword is used only for the TCP protocol to indicate an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which indicate that the packet belongs to an existing connection.

```
interface ethernet 0
 ip access-group 102 in
!
access-list 102 permit tcp any 172.18.0.0 0.0.255.255 established
access-list 102 permit tcp any host 172.18.1.2 eq 25
```

Example Preventing Access to the Web By Filtering on Port Name

In the following example, the Winter and Smith workstations are not allowed web access; other hosts on network 172.20.0.0 are allowed web access:

```
interface ethernet 0
 ip access-group no_web out
!
ip access-list extended no_web
 remark Do not allow Winter to browse the web
 deny host 172.20.3.85 any eq http
 remark Do not allow Smith to browse the web
 deny host 172.20.3.13 any eq http
 remark Allow others on our network to browse the web
 permit 172.20.0.0 0.0.255.255 any eq http
```

Example Filtering on Source Address and Logging the Packets Permitted and Denied

The following example defines access lists 1 and 2, both of which have logging enabled:

```
interface ethernet 0
 ip address 172.16.1.1 255.0.0.0
 ip access-group 1 in
 ip access-group 2 out
!
access-list 1 permit 172.25.0.0 0.0.255.255 log
access-list 1 deny 172.30.0.0 0.0.255.255 log
!
access-list 2 permit 172.27.3.4 log
access-list 2 deny 172.17.0.0 0.0.255.255 log
```

If the interface receives 10 packets from 172.25.7.7 and 14 packets from 172.17.23.21, the first log will look like the following:

```
list 1 permit 172.25.7.7 1 packet
list 2 deny 172.17.23.21 1 packet
```

Five minutes later, the console will receive the following log:

```
list 1 permit 172.25.7.7 9 packets
list 2 deny 172.17.23.21 13 packets
```

Example: Limiting Debug Output

The following sample configuration uses an access list to limit the **debug** command output. Limiting the **debug** output restricts the volume of data to what you are interested in, saving you time and resources.

```
Device(config)# ip access-list acl1
Device(config-std-nacl)# remark Displays only advertisements for LDP peer in acl1
Device(config-std-nacl)# permit host 10.0.0.44

Device# debug mpls ldp advertisements peer-acl acl1

tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.17.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.16.0.31
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.22.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.1
```

```
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.3
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.1.33
```

Where to Go Next

This module describes how to create an access list that permits or denies packets based on source or destination address or protocol. However, there are other fields you could filter on, and other ways to use access lists. If you want to create an access list that filters on other fields or if you want to apply an access list to something other than an interface, you should decide what you want to restrict in your network and determine the type of access list that achieves your goal.

See the following table for references to other fields to filter and other ways to use an IP access list.

If you want to...	See
Filter based on IP Options, TCP flags, noncontiguous ports, or TTL value	“Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values” module
Reorder your access list entries	"Refining an IP Access List” module
Limit access list entries to a time of day or week	"Refining an IP Access List” module
Restrict packets with noninitial fragments	"Refining an IP Access List" module
Restrict access to virtual terminal lines	"Controlling Access to a Virtual Terminal Line”
Control routing updates	“Configuring Routing Protocol-Independent Features” module in the *Cisco IOS IP Routing Protocols Configuration Guide*
Identify or classify traffic for features such as congestion avoidance, congestion management, and priority queuing	“Regulating Packet Flow on a Per-Interface Basis--Using Generic Traffic Shaping” module in the *Quality of Service Solutions Configuration Guide*

Additional References for Creating an IP Access List to Filter TCP Flags

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Related Topic	Document Title
Security Commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
Order of access list entries	"Refining an IP Access List"
Access list entries based on time of day or week	"Refining an IP Access List"
Packets with noninitial fragments	"Refining an IP Access List"
Filtering on IP Options, TCP flags, noncontiguous ports, or TTL values	"Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values"
Access to virtual terminal lines	"Controlling Access to a Virtual Terminal Line"
Routing updates and policy routing	"Configuring Routing Protocol-Independent Features" modules in the *Cisco IOS IP Routing Protocols Configuration Guide*
Traffic identification or classification for features such as congestion avoidance, congestion management, and priority queuing	"Regulating Packet Flow on a Per-Interface Basis--Using Generic Traffic Shaping" module in the *Quality of Service Solutions Configuration Guide*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Creating an IP Access List and Applying It to an Interface

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 7: Feature Information for Creating an IP Access List and Applying It to an Interface

Feature Name	Releases	Feature Configuration Information
Creating an IP Access List and Applying It to an Interface	12.0(32)S4	IP access lists provide many benefits for securing a network and achieving nonsecurity goals, such as determining quality of service (QoS) factors or limiting debug command output. This module describes how to create standard, extended, named, and numbered IP access lists. An access list can be referenced by a name or a number. Standard access lists filter on only the source address in IP packets. Extended access lists can filter on source address, destination address, and other fields in an IP packet.

CHAPTER 7

Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports

This module describes how to use an IP access list to filter IP packets that contain certain IP Options, TCP flags, noncontiguous ports.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Creating an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports

Before you perform any of the tasks in this module, you should be familiar with the information in the following modules:

- "IP Access List Overview"
- "Creating an IP Access List and Applying It to an Interface"

Information About Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports

IP Options

IP uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

The Options, commonly referred to as IP Options, provide for control functions that are required in some situations but unnecessary for the most common communications. IP Options include provisions for time stamps, security, and special routing.

IP Options may or may not appear in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation. In some environments the security option may be required in all datagrams.

The option field is variable in length. There may be zero or more options. IP Options can have one of two formats:

- Format 1: A single octet of option-type.
- Format 2: An option-type octet, an option-length octet, and the actual option-data octets.

The option-length octet counts the option-type octet, the option-length octet, and the option-data octets.

The option-type octet is viewed as having three fields: a 1-bit copied flag, a 2-bit option class, and a 5-bit option number. These fields form an 8-bit value for the option type field. IP Options are commonly referred to by their 8-bit value.

For a complete list and description of IP Options, refer to RFC 791, *Internet Protocol* at the following URL: http://www.faqs.org/rfcs/rfc791.html

Benefits of Filtering IP Options

- Filtering of packets that contain IP Options from the network relieves downstream devices and hosts of the load from options packets.

- This feature also minimizes load to the Route Processor (RP) for packets with IP Options that require RP processing on distributed systems. Previously, the packets were always routed to or processed by the RP CPU. Filtering the packets prevents them from impacting the RP.

Benefits of Filtering on TCP Flags

The ACL TCP Flags Filtering feature provides a flexible mechanism for filtering on TCP flags. Previously, an incoming packet was matched as long as any TCP flag in the packet matched a flag specified in the access control entry (ACE). This behavior allows for a security loophole, because packets with all flags set could get past the access control list (ACL). The ACL TCP Flags Filtering feature allows you to select any combination of flags on which to filter. The ability to match on a flag set and on a flag not set gives you a greater degree of control for filtering on TCP flags, thus enhancing security.

Because TCP packets can be sent as false synchronization packets that can be accepted by a listening port, it is recommended that administrators of firewall devices set up some filtering rules to drop false TCP packets.

The ACEs that make up an access list can be configured to detect and drop unauthorized TCP packets by allowing only the packets that have a very specific group of TCP flags set or not set. The ACL TCP Flags Filtering feature provides a greater degree of packet-filtering control in the following ways:

- You can select any desired combination of TCP flags on which to filter TCP packets.
- You can configure ACEs to allow matching on a flag that is set, as well as on a flag that is not set.

TCP Flags

The table below lists the TCP flags, which are further described in RFC 793, *Transmission Control Protocol*.

Table 8: TCP Flags

TCP Flag	Purpose
ACK	Acknowledge flag—Indicates that the acknowledgment field of a segment specifies the next sequence number the sender of this segment is expecting to receive.
FIN	Finish flag—Used to clear connections.
PSH	Push flag—Indicates the data in the call should be immediately pushed through to the receiving user.
RST	Reset flag—Indicates that the receiver should delete the connection without further interaction.
SYN	Synchronize flag—Used to establish connections.
URG	Urgent flag—Indicates that the urgent field is meaningful and must be added to the segment sequence number.

Benefits of Using the Named ACL Support for Noncontiguous Ports on an Access Control Entry Feature

This feature greatly reduces the number of access control entries (ACEs) required in an access control list to handle multiple entries for the same source address, destination address, and protocol. If you maintain large numbers of ACEs, use this feature to consolidate existing groups of access list entries wherever it is possible and when you create new access list entries. When you configure access list entries with noncontiguous ports, you will have fewer access list entries to maintain.

How Filtering on TTL Value Works

IP extended named and numbered access lists may filter on the TTL value of packets arriving at or leaving an interface. Packets with any possible TTL values 0 through 255 may be permitted or denied (filtered). Like filtering on other fields, such as source or destination address, the **ip access-group** command specifies **in** or **out**, which makes the access list ingress or egress and applies it to incoming or outgoing packets, respectively. The TTL value is checked in conjunction with the specified protocol, application, and any other settings in the access list entry, and all conditions must be met.

Special Handling for Packets with TTL Value of 0 or 1 Arriving at an Ingress Interface

The software switching paths—distributed Cisco Express Forwarding (dCEF), CEF, fast switching, and process switching—will usually permit or discard the packets based on the access list statements. However, when the TTL value of packets arriving at an ingress interface have a TTL of 0 or 1, special handling is required. The packets with a TTL value of 0 or 1 get sent to the process level before the ingress access list is checked in CEF, dCEF, or the fast switching paths. The ingress access list is applied to packets with TTL values 2 through 255 and a permit or deny decision is made.

Packets with a TTL value of 0 or 1 are sent to the process level because they will never be forwarded out of the device; the process level must check whether each packet is destined for the device and whether an Internet Control Message Protocol (ICMP) TTL Expire message needs to be sent back. This means that even if an ACL with TTL value 0 or 1 filtering is configured on the ingress interface with the intention to drop packets with a TTL of 0 or 1, the dropping of the packets will not happen in the faster paths. It will instead happen in the process level when the process applies the ACL. This is also true for hardware switching platforms. Packets with TTL value of 0 or 1 are sent to the process level of the route processor (RP) or Multilayer Switch Feature Card (MSFC).

On egress interfaces, access list filtering on TTL value works just like other access list features. The check will happen in the fastest switching path enabled in the device. This is because the faster switching paths handle all the TTL values (0 through 255) equally on the egress interface.

Control Plane Policing for Filtering TTL Values 0 and 1

The special behavior for packets with a TTL value of 0 or 1 results in higher CPU usage for the device. If you are filtering on TTL value of 0 or 1, you should use control plane policing (CPP) to protect the CPU from being overwhelmed. In order to leverage CPP, you must configure an access list especially for filtering TTL values 0 and 1 and apply the access list through CPP. This access list will be a separate access list from any other interface access lists. Because CPP works for the entire system, not just on individual interfaces, you would need to configure only one such special access list for the entire device. This task is described in the section "Enabling Control Plane Policing to Filter on TTL Values 0 and 1".

Benefits of Filtering on TTL Value

- Filtering on time-to-live (TTL) value provides a way to control which packets are allowed to reach the device or are prevented from reaching the device. By looking at your network layout, you can choose whether to accept or deny packets from a certain device based on how many hops away it is. For example, in a small network, you can deny packets from a location more than three hops away. Filtering on TTL value allows you to validate if the traffic originated from a neighboring device. You can accept only packets that reach you in one hop, for example, by accepting only packets with a TTL value of one less than the initial TTL value of a particular protocol.
- Many control plane protocols communicate only with their neighbors, but receive packets from everyone. By applying an access list that filters on TTL to receiving routers, you can block unwanted packets.
- The Cisco software sends all packets with a TTL value of 0 or 1 to the process level. The device must then send an Internet Control Message Protocol (ICMP) TTL value expire message to the source. By filtering packets that have a TTL value of 0 through 2, you can reduce the load on the process level.

How to Create an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports

Filtering Packets That Contain IP Options

Complete these steps to configure an access list to filter packets that contain IP options and to verify that the access list has been configured correctly.

Note

- The ACL Support for Filtering IP Options feature can be used only with named, extended ACLs.
- Resource Reservation Protocol (RSVP) Multiprotocol Label Switching Traffic Engineering (MPLS TE), Internet Group Management Protocol Version 2 (IGMPV2), and other protocols that use IP options packets may not function in drop or ignore mode if this feature is configured.
- On most Cisco devices, a packet with IP options is not switched in hardware, but requires control plane software processing (primarily because there is a need to process the options and rewrite the IP header), so all IP packets with IP options will be filtered and switched in software.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **deny** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
5. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Repeat Step 4 or Step 5 as necessary.
7. **end**
8. **show ip access-lists** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended mylist1`	Specifies the IP access list by name and enters named access list configuration mode.
Step 4	[*sequence-number*] **deny** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# deny ip any any option traceroute`	(Optional) Specifies a **deny** statement in named IP access list mode. • This access list happens to use a **deny**statement first, but a **permit** statement could appear first, depending on the order of statements you need. • Use the **option** keyword and *option-value* argument to filter packets that contain a particular IP Option. • In this example, any packet that contains the traceroute IP option will be filtered out. • Use the **no** *sequence-number* form of this command to delete an entry.
Step 5	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*	Specifies a **permit** statement in named IP access list mode.

	Command or Action	Purpose
	[**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# permit ip any any option security`	• In this example, any packet (not already filtered) that contains the security IP option will be permitted. • Use the **no** *sequence-number* form of this command to delete an entry.
Step 6	Repeat Step 4 or Step 5 as necessary.	Allows you to revise the access list.

Step 7	**end**
	
	Example:
	`Device(config-ext-nacl)# end`
		(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

Step 8	**show ip access-lists** *access-list-name* **Example:** `Device# show ip access-lists mylist1`	(Optional) Displays the contents of the IP access list.

What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Note To effectively eliminate all packets that contain IP Options, we recommend that you configure the global **ip options drop** command.

Filtering Packets That Contain TCP Flags

This task configures an access list to filter packets that contain TCP flags and verifies that the access list has been configured correctly.

Note

- TCP flag filtering can be used only with named, extended ACLs.
- The ACL TCP Flags Filtering feature is supported only for Cisco ACLs.
- Previously, the following command-line interface (CLI) format could be used to configure a TCP flag-checking mechanism:

permit tcp any any rst The following format that represents the same ACE can now be used: **permit tcp any any match-any +rst** Both the CLI formats are accepted; however, if the new keywords **match-all** or **match-any** are chosen, they must be followed by the new flags that are prefixed with "+" or "**-**". It is advisable to use only the old format or the new format in a single ACL. You cannot mix and match the old and new CLI formats.

Caution If a device having ACEs with the new syntax format is reloaded with a previous version of the Cisco software that does not support the ACL TCP Flags Filtering feature, the ACEs will not be applied, leading to possible security loopholes.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **permit tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
5. [*sequence-number*] **deny tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number*command to delete an entry.
7. **end**
8. **show ip access-lists** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended kmd1`	Specifies the IP access list by name and enters named access list configuration mode.
Step 4	[*sequence-number*] **permit tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# permit tcp any any match-any +rst`	Specifies a **permit** statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • Use the TCP command syntax of the **permit**command. • Any packet with the RST TCP header flag set will be matched and allowed to pass the named access list kmd1 in Step 3.
Step 5	[*sequence-number*] **deny tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# deny tcp any any match-all -ack -fin`	(Optional) Specifies a **deny** statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • Use the TCP command syntax of the **deny**command. • Any packet that does not have the ACK flag set, and also does not have the FIN flag set, will not be allowed to pass the named access list kmd1 in Step 3. • See the **deny**(IP) command for additional command syntax to permit upper-layer protocols (ICMP, IGMP, TCP, and UDP).
Step 6	Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number*command to delete an entry.	Allows you to revise the access list.

Step 7	**end**
	
	Example:
	`Device(config-ext-nacl)# end`
		(Optional) Exits the configuration mode and returns to privileged EXEC mode.

Step 8	**show ip access-lists** *access-list-name*	(Optional) Displays the contents of the IP access list.

	Command or Action	Purpose
	Example: `Device# show ip access-lists kmd1`	• Review the output to confirm that the access list includes the new entry.

What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Configuring an Access Control Entry with Noncontiguous Ports

Perform this task to create access list entries that use noncontiguous TCP or UDP port numbers. Although this task uses TCP ports, you could use the UDP syntax of the **permit** and **deny** commands to filter noncontiguous UDP ports.

Although this task uses a **permit** command first, use the **permit** and **deny** commands in the order that achieves your filtering goals.

Note The ACL—Named ACL Support for Noncontiguous Ports on an Access Control Entry feature can be used only with named, extended ACLs.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **permit tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
5. [*sequence-number*] **deny tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
7. **end**
8. **show ip access-lists** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended acl-extd-1`	Specifies the IP access list by name and enters named access list configuration mode.
Step 4	[*sequence-number*] **permit tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# permit tcp any eq telnet ftp any eq 450 679`	Specifies a **permit** statement in named IP access list configuration mode. • Operators include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range). • If the operator is positioned after the source and source-wildcard arguments, it must match the source port. If the operator is positioned after the destination and destination-wildcard arguments, it must match the destination port. • The **range** operator requires two port numbers. You can configure up to 10 ports after the **eq** and **neq**operators. All other operators require one port number. • To filter UDP ports, use the UDP syntax of this command.
Step 5	[*sequence-number*] **deny tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# deny tcp any neq 45 565 632`	(Optional) Specifies a **deny** statement in named access list configuration mode. • Operators include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range). • If the *operator* is positioned after the *source* and *source-wildcard* arguments, it must match the source port. If the *operator* is positioned after the *destination* and *destination-wildcard* arguments, it must match the destination port. • The **range** operator requires two port numbers. You can configure up to 10 ports after the **eq** and **neq**operators. All other operators require one port number. • To filter UDP ports, use the UDP syntax of this command.

	Command or Action	Purpose
Step 6	Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise the access list.

Step 7	**end**
	
	Example:
	`Device(config-ext-nacl)# end`
		(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

Step 8	**show ip access-lists** *access-list-name* **Example:** `Device# show ip access-lists kmd1`	(Optional) Displays the contents of the access list.

Consolidating Access List Entries with Noncontiguous Ports into One Access List Entry

Perform this task to consolidate a group of access list entries with noncontiguous ports into one access list entry.

Although this task uses TCP ports, you could use the UDP syntax of the **permit** and **deny** commands to filter noncontiguous UDP ports.

Although this task uses a **permit** command first, use the **permit** and **deny** commands in the order that achieves your filtering goals.

SUMMARY STEPS

1. **enable**
2. **show ip access-lists** *access-list-name*
3. **configure terminal**
4. **ip access-list extended** *access-list-name*
5. **no** [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. [*sequence-number*] **permit** *protocol source source-wildcard*[*operator port*[*port*]] *destination destination-wildcard*[*operator port*[*port*]] [**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
7. Repeat Steps 5 and 6 as necessary, adding **permit** or **deny** statements to consolidate access list entries where possible. Use the **no** *sequence-number* command to delete an entry.
8. **end**
9. **show ip access-lists** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip access-lists** *access-list-name* **Example:** `Device# show ip access-lists mylist1`	(Optional) Displays the contents of the IP access list. • Review the output to see if you can consolidate any access list entries.
Step 3	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 4	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended mylist1`	Specifies the IP access list by name and enters named access list configuration mode.
Step 5	**no** [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# no 10`	Removes the redundant access list entry that can be consolidated. • Repeat this step to remove entries to be consolidated because only the port numbers differ. • After this step is repeated to remove the access list entries 20, 30, and 40, for example, those entries are removed because they will be consolidated into one **permit** statement. • If a *sequence-number* is specified, the rest of the command syntax is optional.
Step 6	[*sequence-number*] **permit** *protocol source source-wildcard*[*operator port*[*port*]] *destination destination-wildcard*[*operator port*[*port*]] [**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# permit tcp any neq 45 565 632 any eq 23 45 34 43`	Specifies a **permit** statement in named access list configuration mode. • In this instance, a group of access list entries with noncontiguous ports was consolidated into one **permit** statement. • You can configure up to 10 ports after the **eq** and **neq** operators.
Step 7	Repeat Steps 5 and 6 as necessary, adding **permit** or **deny** statements to consolidate access list entries where possible. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise the access list.

	Command or Action	Purpose

Step 8	**end**
	
	Example:
	`Device(config-std-nacl)# end`
		(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

Step 9	**show ip access-lists** *access-list-name* **Example:** `Device# show ip access-lists mylist1`	(Optional) Displays the contents of the access list.

What To Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Filtering Packets Based on TTL Value

Because access lists are very flexible, it is not possible to define only one combination of **permit** and **deny** commands to filter packets based on the TTL value. This task illustrates just one example that achieves TTL filtering. Configure the appropriate **permit** and **deny** statements that will accomplish your filtering plan.

Note When the access list specifies the operation EQ or NEQ, depending on the Cisco software release in use on the device, the access lists can specify up to ten TTL values. The number of TTL values can vary by the Cisco software release.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**ttl** *operator value*] [**log**] [**time-range** *time-range-name*] [**fragments**]
5. Continue to add **permit** or **deny** statements to achieve the filtering you want.
6. **exit**
7. **interface** *type number*
8. **ip access-group** *access-list-name* {**in** | **out**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended ttlfilter`	Defines an IP access list by name. • An access list that filters on TTL value must be an extended access list.
Step 4	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**ttl** *operator value*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# permit ip host 172.16.1.1 any ttl lt 2`	Sets conditions to allow a packet to pass a named IP access list. • Every access list must have at least one **permit** statement. • This example permits packets from source 172.16.1.1 to any destination with a TTL value less than 2.
Step 5	Continue to add **permit** or **deny** statements to achieve the filtering you want.	--
Step 6	**exit** **Example:** `Device(config-ext-nacl)# exit`	Exits any configuration mode to the next highest mode in the command-line interface (CLI) mode hierarchy.
Step 7	**interface** *type number* **Example:** `Device(config)# interface ethernet 0`	Configures an interface type and enters interface configuration mode.

Step 8	**ip access-group** *access-list-name* {**in** | **out**}
	
	Example:
	
	`Device(config-if)# ip access-group ttlfilter in`
		Applies the access list to an interface.

Enabling Control Plane Policing to Filter on TTL Values 0 and 1

Perform this task to filter IP packets based on a TTL value of 0 or 1 and to protect the CPU from being overwhelmed. This task configures an access list for classification on TTL value 0 and 1, configures the Modular QoS Command-Line Interface (CLI) (MQC), and applies a policy map to the control plane. Any packets that pass the access list are dropped. This special access list is separate from any other interface access lists.

Because access lists are very flexible, it is not possible to define only one combination of **permit** and **deny** commands to filter packets based on the TTL value. This task illustrates just one example that achieves TTL filtering. Configure the appropriate **permit** and **deny** statements that will accomplish your filtering plan.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **ttl** *operator* value
5. Continue to add **permit** or **deny** statements to achieve the filtering you want.
6. **exit**
7. **class-map** *class-map-name* [**match-all** | **match-any**]
8. **match access-group** {*access-group* | **name** *access-group-name*}
9. **exit**
10. **policy-map** *policy-map-name*
11. **class** {*class-name* | **class-default**}
12. **drop**
13. **exit**
14. **exit**
15. **control-plane**
16. **service-policy** {**input** | **output**} *policy-map-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended ttlfilter`	Defines an IP access list by name. • An access list that filters on a TTL value must be an extended access list.
Step 4	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **ttl** *operator* value **Example:** `Device(config-ext-nacl)# permit ip host 172.16.1.1 any ttl lt 2`	Sets conditions to allow a packet to pass a named IP access list. • Every access list must have at least one **permit** statement. • This example permits packets from source 172.16.1.1 to any destination with a TTL value less than 2.
Step 5	Continue to add **permit** or **deny** statements to achieve the filtering you want.	The packets that pass the access list will be dropped.
Step 6	**exit** **Example:** `Device(config-ext-nacl)# exit`	Exits any configuration mode to the next highest mode in the CLI mode hierarchy.

Step 7	**class-map** *class-map-name* [**match-all** | **match-any**]
	
	Example:
	
	`Device(config)# class-map acl-filtering`
		Creates a class map to be used for matching packets to a specified class.

Step 8	**match access-group** {*access-group* | **name** *access-group-name*}
	
	Example:
	
	`Device(config-cmap)# match access-group name ttlfilter`
		Configures the match criteria for a class map on the basis of the specified access control list.

Step 9	**exit** **Example:** `Device(config-cmap)# exit`	Exits any configuration mode to the next highest mode in the CLI mode hierarchy.
Step 10	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map acl-filter`	Creates or modifies a policy map that can be attached to one or more interface to specify a service policy.

	Command or Action	Purpose

Step 11	**class** {*class-name* | **class-default**}
	
	Example:
	
	`Device(config-pmap)# class acl-filter-class`
		Specifies the name of the class whose policy you want to create or change or to specify the default class (commonly known as the class-default class) before you configure its policy.

Step 12	**drop** **Example:** `Device(config-pmap-c)# drop`	Configures a traffic class to discard packets belonging to a specific class.
Step 13	**exit** **Example:** `Device(config-pmap-c)# exit`	Exits any configuration mode to the next highest mode in the CLI mode hierarchy.
Step 14	**exit** **Example:** `Device(config-pmap)# exit`	Exits any configuration mode to the next highest mode in the CLI mode hierarchy.
Step 15	**control-plane** **Example:** `Device(config)# control-plane`	Associates or modifies attributes or parameters that are associated with the control plane of the device.
Step 16	**service-policy** {**input** | **output**} *policy-map-name* **Example:** `Device(config-cp)# service-policy input acl-filter`	Attaches a policy map to a control plane for aggregate control plane services.

Configuration Examples for Filtering IP Options, TCP Flags, Noncontiguous Ports

Example: Filtering Packets That Contain IP Options

The following example shows an extended access list named mylist2 that contains access list entries (ACEs) that are configured to permit TCP packets only if they contain the IP Options that are specified in the ACEs:

```
ip access-list extended mylist2
 10 permit ip any any option eool
 20 permit ip any any option record-route
 30 permit ip any any option zsu
 40 permit ip any any option mtup
```

The **show access-list** command has been entered to show how many packets were matched and therefore permitted:

```
Device# show ip access-list mylist2
Extended IP access list test
10 permit ip any any option eool (1 match)
20 permit ip any any option record-route (1 match)
30 permit ip any any option zsu (1 match)
40 permit ip any any option mtup (1 match)
```

Example: Filtering Packets That Contain TCP Flags

The following access list allows TCP packets only if the TCP flags ACK and SYN are set and the FIN flag is not set:

```
ip access-list extended aaa
 permit tcp any any match-all +ack +syn -fin
 end
```

The **show access-list** command has been entered to display the ACL:

```
Device# show access-list aaa

Extended IP access list aaa
 10 permit tcp any any match-all +ack +syn -fin
```

Example: Creating an Access List Entry with Noncontiguous Ports

The following access list entry can be created because up to ten ports can be entered after the **eq** and **neq** operators:

```
ip access-list extended aaa
 permit tcp any eq telnet ftp any eq 23 45 34
 end
```

Enter the **show access-lists** command to display the newly created access list entry.

```
Device# show access-lists aaa
```

```
Extended IP access list aaa
 10 permit tcp any eq telnet ftp any eq 23 45 34
```

Example: Consolidating Some Existing Access List Entries into One Access List Entry with Noncontiguous Ports

The **show access-lists** command is used to display a group of access list entries for the access list named abc:

```
Device# show access-lists abc
Extended IP access list abc
 10 permit tcp any eq telnet any eq 450
 20 permit tcp any eq telnet any eq 679
 30 permit tcp any eq ftp any eq 450
 40 permit tcp any eq ftp any eq 679
```

Because the entries are all for the same **permit** statement and simply show different ports, they can be consolidated into one new access list entry. The following example shows the removal of the redundant access list entries and the creation of a new access list entry that consolidates the previously displayed group of access list entries:

```
ip access-list extended abc
 no 10
 no 20
 no 30
 no 40
 permit tcp any eq telnet ftp any eq 450 679
 end
```

When the **show access-lists** command is reentered, the consolidated access list entry is displayed:

```
Device# show access-lists abc
Extended IP access list abc
 10 permit tcp any eq telnet ftp any eq 450 679
```

Example: Filtering on TTL Value

The following access list filters IP packets containing type of service (ToS) level 3 with time-to-live (TTL) values 10 and 20. It also filters IP packets with a TTL greater than 154 and applies that rule to noninitial fragments. It permits IP packets with a precedence level of flash and a TTL value not equal to 1, and it sends log messages about such packets to the console. All other packets are denied.

```
ip access-list extended incomingfilter
 deny ip any any tos 3 ttl eq 10 20
 deny ip any any ttl gt 154 fragments
 permit ip any any precedence flash ttl neq 1 log
!
interface ethernet 0

ip access-group incomingfilter in
```

Example: Control Plane Policing to Filter on TTL Values 0 and 1

The following example configures a traffic class called acl-filter-class for use in a policy map called acl-filter. An access list permits IP packets from any source having a time-to-live (TTL) value of 0 or 1. Any packets matching the access list are dropped. The policy map is attached to the control plane.

```
ip access-list extended ttlfilter
 permit ip any any ttl eq 0 1
class-map acl-filter-class

 match access-group name ttlfilter

policy-map acl-filter
 class acl-filter-class
 drop
control-plane
 service-policy input acl-filter
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Security commands	*Cisco IOS Security Command Reference*
Configuring the device to drop or ignore packets containing IP Options by using the **no ip options** command.	"ACL IP Options Selective Drop"
Overview information about access lists.	"IP Access List Overview"
Information about creating an IP access list and applying it to an interface	"Creating an IP Access List and Applying It to an Interface"
QoS commands	*Cisco IOS Quality of Service Solutions Command Reference*

RFCs

RFC	Title
RFC 791	*Internet Protocol* http://www.faqs.org/rfcs/rfc791.html http://www.faqs.org/rfcs/rfc791.html
RFC 793	*Transmission Control Protocol*
RFC 1393	*Traceroute Using an IP Option*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Creating an IP Access List to Filter

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 9: Feature Information for Creating an IP Access List to Filter

Feature Name	Releases	Feature Configuration Information
ACL--Named ACL Support for Noncontiguous Ports on an Access Control Entry	12.3(7)T 12.2(25)S	This feature allows you to specify noncontiguous ports in a single access control entry, which greatly reduces the number of entries required in an access control list when several entries have the same source address, destination address, and protocol, but differ only in the ports.

Feature Name	Releases	Feature Configuration Information
ACL Support for Filtering IP Options	12.3(4)T 12.2(25)S 15.2(2)S 15.4(1)S	This feature allows you to filter packets having IP Options, in order to prevent routers from becoming saturated with spurious packets. In Cisco IOS Release 15.4(1)S, support was added for the Cisco ASR 901S series routers.
ACL TCP Flags Filtering	12.3(4)T 12.2(25)S	This feature provides a flexible mechanism for filtering on TCP flags. Before Cisco IOS Release 12.3(4)T, an incoming packet was matched as long as any TCP flag in the packet matched a flag specified in the access control entry (ACE). This behavior allows for a security loophole, because packets with all flags set could get past the access control list (ACL). The ACL TCP Flags Filtering feature allows you to select any combination of flags on which to filter. The ability to match on a flag set and on a flag not set gives you a greater degree of control for filtering on TCP flags, thus enhancing security.

CHAPTER 8

ACL Syslog Correlation

The Access Control List (ACL) Syslog Correlation feature appends a tag (either a user-defined cookie or a device-generated MD5 hash value) to access control entry (ACE) syslog entries. This tag uniquely identifies the ACE , within the ACL, that generated the syslog entry.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for ACL Syslog Correlation

Before you configure the ACL Syslog Correlation feature, you must understand the concepts in the "IP Access List Overview" module.

The ACL Syslog Correlation feature appends a user-defined cookie or a device-generated hash value to ACE messages in the syslog. These values are only appended to ACE messages when the log option is enabled for the ACE.

Information About ACL Syslog Correlation

ACL Syslog Correlation Tags

The ACL Syslog Correlation feature appends a tag (either a user-defined cookie or a device-generated MD5 hash value) to access control entry (ACE) syslog entries. This tag uniquely identifies an ACE that generated the syslog entry.

Network management software can use the tag to identify which ACE generated a specific syslog event. For example, network administrators can select an ACE rule in the network management application and can then view the corresponding syslog events for that ACE rule.

To append a tag to the syslog message, the ACE that generates the syslog event must have the log option enabled. The system appends only one type of tag (either a user-defined cookie or a device-generated MD5 hash value) to each message.

To specify a user-defined cookie tag, the user must enter the cookie value when configuring the ACE log option. The cookie must be in alpha-numeric form, it cannot be greater than 64 characters, and it cannot start with hex-decimal notation (such as 0x).

To specify a device-generated MD5 hash value tag, the hash-generation mechanism must be enabled on the device and the user must not enter a cookie value while configuring the ACE log option.

ACE Syslog Messages

When a packet is matched against an access control entry (ACE) in an ACL, the system checks whether the log option is enabled for that event. If the log option is enabled and the ACL Syslog Correlation feature is configured on the device, the system attaches the tag to the syslog message. The tag is displayed at the end of the syslog message, in addition to the standard information.

The following is a sample syslog message showing a user-defined cookie tag:

```
Jun 5 12:55:44.359: %SEC-6-IPACCESSLOGP: list logacl permitted tcp 192.168.16.1(38402) ->
192.168.16.2(23), 1 packet [User_permiited_ACE]
```

The following is a sample syslog message showing a hash value tag:

```
Jun 5 12:55:44.359: %SEC-6-IPACCESSLOGP: list logacl permitted tcp 192.168.16.1(38402) ->
192.168.16.2(23), 1 packet [0x723E6E12]
```

How to Configure ACL Syslog Correlation

Enabling Hash Value Generation on a Device

Perform this task to configure the device to generate an MD5 hash value for each log-enabled access control entry (ACE) in the system that is not configured with a user-defined cookie.

When the hash value generation setting is enabled, the system checks all existing ACEs and generates a hash value for each ACE that requires one. When the hash value generation setting is disabled, all previously generated hash values are removed from the system.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list logging hash-generation**
4. **end**
5. Do one of the following:
    - **show ip access-list** *access-list-number*
    - **show ip access-list** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list logging hash-generation** **Example:** `Device(config)# ip access-list logging hash-generation`	Enables hash value generation on the device. • If an ACE exists that is log enabled, and requires a hash value, the device automatically generates the value and displays the value on the console.

Step 4	**end**
	
	Example:
	
	`Device(config)# end`
		(Optional) Exits global configuration mode and returns to privileged EXEC mode.

Step 5	Do one of the following: • **show ip access-list** *access-list-number* • **show ip access-list** *access-list-name*	(Optional) Displays the contents of the numbered or named IP access list. • Review the output to confirm that the access list for a log-enabled ACE includes the generated hash value.

	Command or Action	Purpose

	Example:
	`Device# show ip access-list 101`
	Example:
	`Device# show ip access-list acl`	

Disabling Hash Value Generation on a Device

Perform this task to disable hash value generation on the device. When the hash value generation setting is disabled, all previously generated hash values are removed from the system.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no ip access-list logging hash-generation**
4. **end**
5. Do one of the following:
    - **show ip access-list** *access-list-number*
    - **show ip access-list** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no ip access-list logging hash-generation**	Disables hash value generation on the device.

	Command or Action	Purpose
	Example: `Device(config)# no ip access-list logging hash-generation`	• The system removes any previously created hash values from the system.

Step 4	**end**
	Example:
	`Device(config)# end`
		(Optional) Exits global configuration mode and returns to privileged EXEC mode.

Step 5	Do one of the following:
	• **show ip access-list** *access-list-number*
	• **show ip access-list** *access-list-name*
	Example:
	`Device# show ip access-list 101`
	Example:
	`Device# show ip access-list acl`
		(Optional) Displays the contents of the IP access list.
		• Review the output to confirm that the access list for a log-enabled ACE does not have a generated hash value.

Configuring ACL Syslog Correlation Using a User-Defined Cookie

Perform this task to configure the ACL Syslog Correlation feature on a device for a specific access list, using a user-defined cookie as the syslog message tag.

The example in this section shows how to configure the ACL Syslog Correlation feature using a user-defined cookie for a numbered access list. However, you can configure the ACL Syslog Correlation feature using a user-defined cookie for both numbered and named access lists, and for both standard and extended access lists.

Note The following restrictions apply when choosing the user-defined cookie value:

- The maximum number of characters is 64.
- The cookie cannot start with hexadecimal notation (such as 0x).
- The cookie cannot be the same as, or a subset of, the following keywords: **reflect**, **fragment**, **time-range**. For example, reflect and ref are not valid values. However, the cookie can start with the keywords. For example, reflectedACE and fragment_33 are valid values
- The cookie must contains only alphanumeric characters.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* **permit** *protocol source destination* **log** *word*
4. **end**
5. **show ip access-list** *access-list-number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**access-list** *access-list-number* **permit** *protocol source destination* **log** *word* **Example:** `Device(config)# access-list 101 permit tcp host 10.1.1.1 host 10.1.1.2 log UserDefinedValue`	Defines an extended IP access list and a user-defined cookie value. • Enter the cookie value as the *word*argument.

Step 4	**end**
	
	Example:
	`Device(config)# end`
		(Optional) Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 5	**show ip access-list** *access-list-number* **Example:** `Device# show ip access-list 101`	(Optional) Displays the contents of the IP access list. • Review the output to confirm that the access list includes the user-defined cookie value.

Examples

The following is sample output from the **show ip access-list** command for an access list with a user-defined cookie value.

```
Device# show ip access-list
101
Extended IP access list 101
30 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = UserDefinedValue)
```

Configuring ACL Syslog Correlation Using a Hash Value

Perform this task to configure the ACL Syslog Correlation feature on a device for a specific access list, using a device-generated hash value as the syslog message tag.

The steps in this section shows how to configure the ACL Syslog Correlation feature using a device-generated hash value for a numbered access list. However, you can configure the ACL Syslog Correlation feature using a device-generated hash value for both numbered and named access lists, and for both standard and extended access lists.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list logging hash-generation**
4. access-list *access-list-number* **permit** *protocol source destination* **log**
5. **end**
6. **show ip access-list** *access-list-number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list logging hash-generation** **Example:** `Device(config)# ip access-list logging hash-generation`	Enables hash value generation on the device. • If an ACE exists that is log enabled, and requires a hash value, the device automatically generates the value and displays the value on the console.
Step 4	access-list *access-list-number* **permit** *protocol source destination* **log** **Example:** `Device(config)# access-list 102 permit tcp host 10.1.1.1 host 10.1.1.2 log`	Defines an extended IP access list. • Enable the log option for the access list, but do not specify a cookie value. • The device automatically generates a hash value for the newly defined access list.

Step 5	**end**
	
	Example:
	
	`Device(config)# end`
		(Optional) Exits global configuration mode and returns to privileged EXEC mode.

Step 6	**show ip access-list** *access-list-number* **Example:** `Device# show ip access-list 102`	(Optional) Displays the contents of the IP access list. • Review the output to confirm that the access list includes the router-generated hash value.

Examples

The following is sample output from the **show ip access-list** command for an access list with a device-generated hash value.

```
Device# show ip access-list
102
Extended IP access list 102
10 permit tcp host 10.1.1.1 host 10.1.1.2 log (hash = 0x7F9CF6B9)
```

Changing the ACL Syslog Correlation Tag Value

Perform this task to change the value of the user-defined cookie or replace a device-generated hash value with a user-defined cookie.

The steps in this section shows how to change the ACL Syslog Correlation tag value on a numbered access list. However, you can change the ACL Syslog Correlation tag value for both numbered and named access lists, and for both standard and extended access lists.

SUMMARY STEPS

1. **enable**
2. show access-list
3. **configure terminal**
4. access-list *access-list-number* **permit** *protocol source destination* **log** *word*
5. **end**
6. **show ip access-list** *access-list-number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	show access-list **Example:** `Device(config)# show access-list`	(Optional) Displays the contents of the access list.
Step 3	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 4	access-list *access-list-number* **permit** *protocol source destination* **log** *word*
	
	Example:
	
	`Device(config)# access-list 101 permit tcp host 10.1.1.1 host 10.1.1.2 log NewUDV`
	
	Example:
	
	`OR`
	
	Example:
		Modifies the cookie or changes the hash value to a cookie.
		• You must enter the entire access list configuration command, replacing the previous tag value with the new tag value.

	Command or Action	Purpose
	Example: `Device(config)# access-list 101 permit tcp any any log replacehash`	

Step 5	**end**
	Example:
	`Device(config)# end`
		(Optional) Exits global configuration mode and returns to privileged EXEC mode.

Step 6	**show ip access-list** *access-list-number* **Example:** `Device# show ip access-list 101`	(Optional) Displays the contents of the IP access list. • Review the output to confirm the changes.

Troubleshooting Tips

Use the **debug ip access-list hash-generation** command to display access list debug information. The following is an example of the **debug** command output:

```
Device# debug ip access-list hash-generation
 Syslog hash code generation debugging is on
Device# show debug
IP ACL:
 Syslog hash code generation debugging is on
Device# no debug ip access-list hash-generation

 Syslog hash code generation debugging is off
Device# show debug
Device#
```

Configuration Examples for ACL Syslog Correlation

Example: Enabling Hash Value Generation on a Device

The following is sample output from the **show ip access-list** command when hash generation is enabled for the specified access-list.

```
Device# show ip access-list 101
Extended IP access list 101
10 permit tcp any any log (hash = 0x75F078B9)
Device# show ip access-list acl
Extended IP access list acl
10 permit tcp any any log (hash = 0x3027EB26)
```

Example: Disabling Hash Value Generation on a Device

The following is sample output from the **show ip access-list** command when hash generation is disabled and no cookie value has been specified.

```
Device# show ip access-list
101
Extended IP access list 101
10 permit tcp any any log
Device# show ip access-list
acl
Extended IP access list acl
10 permit tcp any any log
```

Example: Configuring ACL Syslog Correlation Using a User-Defined Cookie

The following example shows how to configure the ACL Syslog Correlation feature on a device using a user-defined cookie.

```
Device#
Device# debug ip access-list hash-generation
Syslog MD5 hash code generation debugging is on
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# access-list 33 permit 10.10.10.6 log cook_33_std
Device(config)# do show ip access 33
Standard IP access list 33
10 permit 10.10.10.6 log (tag = cook_33_std)
Device(config)# end
```

Example: Configuring ACL Syslog Correlation using a Hash Value

The following examples shows how to configure the ACL Syslog Correlation feature on a device using a device-generated hash value.

```
Device# debug ip access-list hash-generation
Syslog MD5 hash code generation debugging is on
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# access-list 33 permit 10.10.10.7 log
Device(config)#
*Nov 7 13:51:23.615: %IPACL-HASHGEN: Hash Input: 33 standard permit 10.10.10.7
Hash Output: 0xCE87F535
Device(config)#
do show ip access 33

Standard IP access list 33
    10 permit 10.10.10.6 log (tag = cook_33_std)
    20 permit 10.10.10.7 log (hash = 0xCE87F535)
```

Example: Changing the ACL Syslog Correlation Tag Value

The following example shows how to replace an existing access list user-defined cookie with a new cookie value, and how to replace a device-generated hash value with a user-defined cookie value.

```
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# do show ip access-list 101
Extended IP access list 101
    10 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = MyCookie)
    20 permit tcp any any log (hash = 0x75F078B9)
Device(config)# access-list 101 permit tcp host 10.1.1.1 host 10.1.1.2 log NewUDV
Device(config)# do show access-list
Extended IP access list 101
    10 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = NewUDV)
    20 permit tcp any any log (hash = 0x75F078B9)
Device(config)# access-list 101 permit tcp any any log replacehash
Device(config)# do show access-list
Extended IP access list 101
    10 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = NewUDV)
    20 permit tcp any any log (tag = replacehash)
```

Additional References for ACL Syslog Correlation

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
ACL commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
Configuring and Creating ACLs	"Creating an IP Access List and Applying it to an Interface"

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for ACL Syslog Correlation

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 10: Feature Information for ACL Syslog Correlation

Feature Name	Releases	Feature Information
ACL Syslog Correlation		The ACL Syslog Correlation feature appends a tag (either a user-defined cookie or a device-generated MD5 hash value) to ACE syslog entries. This tag uniquely identifies the ACE , within the ACL, that generated the syslog entry. The following commands were introduced or modified: **ip access-list logging hash-generation**, **debug ip access-list hash-generation**, **access-list (IP extended)**, **access-list (IP standard)**, **permit**, **permit (Catalyst 6500 series switches)**, **permit (IP)**.

CHAPTER 9

Refining an IP Access List

There are several ways to refine an access list while or after you create it. You can change the order of the entries in an access list or add entries to an access list. You can restrict access list entries to a certain time of day or week, or achieve finer granularity when filtering packets by filtering noninitial fragments of packets.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About Refining an IP Access List

Access List Sequence Numbers

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

Sequence numbers allow users to add access list entries and resequence them. When you add a new entry, you specify the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Benefits of Access List Sequence Numbers

An access list sequence number is a number at the beginning of a **permit** or **deny** command in an access list. The sequence number determines the order that the entry appears in the access list. The ability to apply sequence numbers to IP access list entries simplifies access list changes.

Prior to having sequence numbers, users could only add access list entries to the end of an access list; therefore, needing to add statements anywhere except the end of the list required reconfiguring the entire access list. There was no way to specify the position of an entry within an access list. If a user wanted to insert an entry (statement) in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When a user adds a new entry, the user chooses the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry. Sequence numbers make revising an access list much easier.

Sequence Numbering Behavior

- For backward compatibility with previous releases, if entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483647. If the generated sequence number exceeds this maximum number, the following message is displayed:

```
Exceeded maximum sequence number.
```

- If the user enters an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- If the user enters an entry that matches an already existing entry (except for the sequence number), then no changes are made.
- If the user enters a sequence number that is already present, the following error message is generated:

```
Duplicate sequence number.
```

- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Distributed support is provided so that the sequence numbers of entries in the Route Processor (RP) and line card are in synchronization at all times.
- Sequence numbers are not nvgened. That is, the sequence numbers themselves are not saved. In the event that the system is reloaded, the configured sequence numbers revert to the default sequence starting

number and increment. The function is provided for backward compatibility with software releases that do not support sequence numbering.

- This feature works with named and numbered, standard and extended IP access lists.

Benefits of Time Ranges

Benefits and possible uses of time ranges include the following:

- The network administrator has more control over permitting or denying a user access to resources. These resources could be an application (identified by an IP address/mask pair and a port number), policy routing, or an on-demand link (identified as interesting traffic to the dialer).
- Network administrators can set time-based security policy, including the following:
    - Perimeter security using the Cisco IOS Firewall feature set or access lists
    - Data confidentiality with Cisco Encryption Technology or IP Security Protocol (IPSec)
- Policy-based routing (PBR) and queueing functions are enhanced.
- When provider access rates vary by time of day, it is possible to automatically reroute traffic cost effectively.
- Service providers can dynamically change a committed access rate (CAR) configuration to support the quality of service (QoS) service level agreements (SLAs) that are negotiated for certain times of day.
- Network administrators can control logging messages. Access list entries can log traffic at certain times of the day, but not constantly. Therefore, administrators can simply deny access without needing to analyze many logs generated during peak hours.

Distributed Time-Based Access Lists

Before the introduction of the Distributed Time-Based Access Lists feature, time-based access lists were not supported on line cards for the Cisco 7500 series routers. If time-based access lists were configured, they behaved as normal access lists. If an interface on a line card were configured with a time-based access list, the packets switched into the interface were not distributed switched through the line card, but were forwarded to the Route Processor for processing.

The Distributed Time-Based Access Lists feature allows packets destined for an interface configured with a time-based access list to be distributed switched through the line card.

For this functionality to work, the software clock must remain synchronized between the Route Processor and the line card. This synchronization occurs through an exchange of interprocess communications (IPC) messages from the Route Processor to the line card. When a time range or a time-range entry is changed, added, or deleted, an IPC message is sent by the Route Processor to the line card.

There is no difference between how the user configures a time-based access list and a distributed time-based access list.

Benefits of Filtering Noninitial Fragments of Packets

If the **fragments**keyword is used in additional IP access list entries that deny fragments, the fragment control feature provides the following benefits:

Additional Security

You are able to block more of the traffic you intended to block, not just the initial fragment of such packets. The unwanted fragments no longer linger at the receiver until the reassembly timeout is reached because they are blocked before being sent to the receiver. Blocking a greater portion of unwanted traffic improves security and reduces the risk from potential hackers.

Reduced Cost

By blocking unwanted noninitial fragments of packets, you are not paying for traffic you intended to block.

Reduced Storage

By blocking unwanted noninitial fragments of packets from ever reaching the receiver, that destination does not have to store the fragments until the reassembly timeout period is reached.

Expected Behavior Is Achieved

The noninitial fragments will be handled in the same way as the initial fragment, which is what you would expect. There are fewer unexpected policy routing results and fewer fragments of packets being routed when they should not be.

Access List Processing of Fragments

The behavior of access list entries regarding the use or lack of use of the **fragments** keyword can be summarized as follows:

If the Access-List Entry Has...	Then...
...no **fragments** keyword (the default), and assuming all of the access-list entry information matches,	For an access list entry that contains only Layer 3 information: • The entry is applied to nonfragmented packets, initial fragments, and noninitial fragments. For an access list entry that contains Layer 3 and Layer 4 information: • The entry is applied to nonfragmented packets and initial fragments. • If the entry is a **permit** statement, then the packet or fragment is permitted. • If the entry is a **deny** statement, then the packet or fragment is denied. • The entry is also applied to noninitial fragments in the following manner. Because noninitial fragments contain only Layer 3 information, only the Layer 3 portion of an access list entry can be applied. If the Layer 3 portion of the access list entry matches, and • If the entry is a **permit** statement, then the noninitial fragment is permitted. • If the entry is a **deny** statement, then the next access list entry is processed. **Note** The **deny** statements are handled differently for noninitial fragments versus nonfragmented or initial fragments.
...the **fragments** keyword, and assuming all of the access-list entry information matches,	The access list entry is applied only to noninitial fragments. The **fragments** keyword cannot be configured for an access list entry that contains any Layer 4 information.

Be aware that you should not add the **fragments** keyword to every access list entry because the first fragment of the IP packet is considered a nonfragment and is treated independently of the subsequent fragments. An initial fragment will not match an access list **permit** or **deny** entry that contains the **fragments** keyword. The packet is compared to the next access list entry, and so on, until it is either permitted or denied by an access list entry that does not contain the **fragments** keyword. Therefore, you may need two access list entries for every **deny** entry. The first **deny** entry of the pair will not include the **fragments** keyword and applies to the initial fragment. The second **deny** entry of the pair will include the **fragments** keyword and applies to the subsequent fragments. In the cases in which there are multiple **deny** entries for the same host but with different Layer 4 ports, a single **deny** access list entry with the **fragments** keyword for that host is all that needs to be added. Thus all the fragments of a packet are handled in the same manner by the access list.

Packet fragments of IP datagrams are considered individual packets, and each counts individually as a packet in access list accounting and access list violation counts.

How to Refine an IP Access List

The tasks in this module provide you with various ways to refine an access list if you did not already do so while you were creating it. You can change the order of the entries in an access list, add entries to an access list, restrict access list entries to a certain time of day or week, or achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

Revising an Access List Using Sequence Numbers

Perform this task if you want to add entries to an existing access list, change the order of entries, or simply number the entries in an access list to accommodate future changes.

Note Remember that if you want to delete an entry from an access list, you can simply use the **no deny** or **no permit** form of the command, or the **no** *sequence-number* command if the statement already has a sequence number.

Note Access list sequence numbers do not support dynamic, reflexive, or firewall access lists.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list resequence** *access-list-name starting-sequence-number increment*
4. **ip access-list** {**standard**| **extended**} *access-list-name*
5. Do one of the following:
   - *sequence-number* **permit** *source source-wildcard*
   - *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Do one of the following:
   - *sequence-number* **deny** *source source-wildcard*
   - *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
7. Repeat Step 5 and Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
8. **end**
9. **show ip access-lists** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list resequence** *access-list-name starting-sequence-number increment* **Example:** `Router(config)# ip access-list resequence kmd1 100 15`	Resequences the specified IP access list using the starting sequence number and the increment of sequence numbers. • This example resequences an access list named kmd1. The starting sequence number is 100 and the increment is 15.
Step 4	**ip access-list** {**standard**| **extended**} *access-list-name*	Specifies the IP access list by name and enters named access list configuration mode.

	Command or Action	Purpose
	Example: `Router(config)# ip access-list standard xyz123`	• If you specify **standard**, make sure you specify subsequent **permit** and **deny** statements using the standard access list syntax. • If you specify **extended**, make sure you specify subsequent **permit** and **deny** statements using the extended access list syntax.
Step 5	Do one of the following: • *sequence-number* **permit** *source source-wildcard* • *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config-std-nacl)# 105 permit 10.5.5.5 0.0.0.255`	Specifies a permit statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • See the **permit** (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP). • Use the **no** *sequence-number* command to delete an entry. • As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Router(config-ext-nacl)# and you would use the extended **permit** command syntax.
Step 6	Do one of the following: • *sequence-number* **deny** *source source-wildcard* • *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config-std-nacl)# 110 deny 10.6.6.7 0.0.0.255`	(Optional) Specifies a deny statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • See the **deny** (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP). • Use the **no** *sequence-number* command to delete an entry. • As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Router(config-ext-nacl)# and you would use the extended **deny** command syntax.
Step 7	Repeat Step 5 and Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise the access list.

Step 8	**end**
	Example:
	`Router(config-std-nacl)# end`
		(Optional) Exits the configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 9	**show ip access-lists** *access-list-name* **Example:** `Router# show ip access-lists xyz123`	(Optional) Displays the contents of the IP access list. • Review the output to see that the access list includes the new entry.

Examples

The following is sample output from the **show ip access-lists** command when the **xyz123** access list is specified.

```
Router# show ip access-lists xyz123
Standard IP access list xyz123
100 permit 10.4.4.0, wildcard bits 0.0.0.255
105 permit 10.5.5.5, wildcard bits 0.0.0.255
115 permit 10.0.0.0, wildcard bits 0.0.0.255
130 permit 10.5.5.0, wildcard bits 0.0.0.255
145 permit 10.0.0.0, wildcard bits 0.0.0.255
```

Restricting an Access List Entry to a Time of Day or Week

By default, access list statements are always in effect once they are applied. However, you can define the times of the day or week that **permit** or **deny** statements are in effect by defining a time range, and then referencing the time range by name in an individual access list statement. IP and Internetwork Packet Exchange (IPX) named or numbered extended access lists can use time ranges.

Before You Begin

The time range relies on the software clock of the routing device. For the time range feature to work the way you intend, you need a reliable clock source. We recommend that you use Network Time Protocol (NTP) to synchronize the software clock of the routing device.

Note The Distributed Time-Based Access Lists feature is supported on Cisco 7500 series routers with a Versatile Interface Processor (VIP) enabled.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **time-range** *time-range-name*
4. **periodic** *days-of-the-week hh* **:** *mm* **to** [*days-of-the-week*] *hh* **:** *mm*
5. Repeat Step 4 if you want more than one period of time applied to an access list statement.
6. **absolute** [**start** *time date*] [**end** *time date*]
7. **exit**
8. Repeat Steps 3 through 7 if you want different time ranges to apply to **permit** or **deny** statements.
9. **ip access-list extended** *name*
10. **deny** *protocol source* [*source-wildcard*] *destination*[*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] **time-range** *time-range-name*
11. **permit** *protocol source* [*source-wildcard*] *destination*[*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] **time-range** *time-range-name*
12. Optionally repeat some combination of Steps 10 and 11 until you have specified the values on which you want to base your access list.
13. **end**
14. **show ip access-list**
15. **show time-range**
16. **show time-range ipc**
17. **clear time-range ipc**
18. **debug time-range ipc**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**time-range** *time-range-name* **Example:** `Router(config)# time-range limit_http`	Defines a time range and enters time-range configuration mode. • The name cannot contain a space or quotation mark, and must begin with a letter. • Multiple time ranges can occur in a single access list.

	Command or Action	Purpose
Step 4	**periodic** *days-of-the-week hh* **:** *mm* **to** [*days-of-the-week*] *hh* **:** *mm* **Example:** `Router(config-time-range)# periodic Monday 6:00 to Wednesday 19:00`	(Optional) Specifies a recurring (weekly) time range. • The first occurrence of *days-of-the-week* is the starting day or day of the week that the associated time range is in effect. The second occurrence is the ending day or day of the week the associated statement is in effect. • The *days-of-the-week*argument can be any single day or combinations of days: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. Other possible values are: • **daily**--Monday through Sunday • **weekdays**--Monday through Friday • **weekend**--Saturday and Sunday • If the ending days of the week are the same as the starting days of the week, they can be omitted. • The first occurrence of *hh:mm* is the starting hours:minutes that the associated time range is in effect. The second occurrence is the ending hours:minutes the associated statement is in effect. • The hours:minutes are expressed in a 24-hour clock. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m.
Step 5	Repeat Step 4 if you want more than one period of time applied to an access list statement.	(Optional) Multiple **periodic** commands are allowed in a time range.
Step 6	**absolute** [**start** *time date*] [**end** *time date*] **Example:** `Router(config-time-range)# absolute start 6:00 1 August 2005 end 18:00 31 October 2005`	(Optional) Specifies an absolute time when a time range is in effect. • Only one **absolute** command is allowed in a time range. • The time is expressed in 24-hour notation, in the form of hours:minutes. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m. The date is expressed in the format *day month year*. The minimum start is 00:00 1 January 1993. If no start time and date are specified, the **permit** or **deny** statement is in effect immediately. • Absolute time and date that the **permit** or **deny** statement of the associated access list is no longer in effect. Same time and date format as described for the **start** keyword. The end time and date must be after the start time and date. The maximum end time is 23:59 31 December 2035. If no end time and date are specified, the associated **permit** or **deny** statement is in effect indefinitely.
Step 7	**exit** **Example:** `Router(config-time-range)# exit`	Exits to the next highest mode.

	Command or Action	Purpose
Step 8	Repeat Steps 3 through 7 if you want different time ranges to apply to **permit** or **deny** statements.	--
Step 9	**ip access-list extended** *name* **Example:** `Router(config)# ip access-list extended autumn`	Defines an extended IP access list using a name and enters extended named access list configuration mode.
Step 10	**deny** *protocol source* [*source-wildcard*] *destination*[*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] **time-range** *time-range-name* **Example:** `Router(config-ext-nacl)# deny tcp 172.16.22.23 any eq http time-range limit_http`	(Optional) Denies any packet that matches all of the conditions specified in the statement. • Specify the time range you created in Step 3. • In this example, one host is denied HTTP access during the time defined by the time range called "limit_http."
Step 11	**permit** *protocol source* [*source-wildcard*] *destination*[*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] **time-range** *time-range-name* **Example:** `Router(config-ext-nacl)# permit tcp any any eq http time-range limit_http`	Permits any packet that matches all of the conditions specified in the statement. • You can specify the time range you created in Step 3 or in a different instance of Step 3, depending on whether you want the time ranges for your statements to be the same or different. • In this example, all other sources are given access to HTTP during the time defined by the time range called "limit_http."
Step 12	Optionally repeat some combination of Steps 10 and 11 until you have specified the values on which you want to base your access list.	--

Step 13	**end**
	
	Example:
	
	`Router(config-ext-nacl)# end`
		Ends configuration mode and returns the system to privileged EXEC mode.

Step 14	**show ip access-list** **Example:** `Router# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

	Command or Action	Purpose
Step 15	**show time-range** **Example:** `Router# show time-range`	(Optional) Displays the time ranges that are set.
Step 16	**show time-range ipc** **Example:** `Router# show time-range ipc`	(Optional) Displays the statistics about the time-range IPC messages between the Route Processor and line card on the Cisco 7500 series router.
Step 17	**clear time-range ipc** **Example:** `Router# clear time-range ipc`	(Optional) Clears the time-range IPC message statistics and counters between the Route Processor and line card on the Cisco 7500 series router.
Step 18	**debug time-range ipc** **Example:** `Router# debug time-range ipc`	(Optional) Enables debugging output for monitoring the time-range IPC messages between the Route Processor and line card on the Cisco 7500 series router.

What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Filtering Noninitial Fragments of Packets

Filter noninitial fragments of packets with an extended access list if you want to block more of the traffic you intended to block, not just the initial fragment of such packets. You should first understand the following concepts.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *name*
4. [*sequence-number*] **deny** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]
5. [*sequence-number*] **deny** *protocol source*[*source-wildcard*][*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **fragments**
6. [*sequence-number*] **permit** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]
7. Repeat some combination of Steps 4 through 6 until you have specified the values on which you want to base your access list.
8. **end**
9. **show ip access-list**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *name* **Example:** `Router(config)# ip access-list extended rstrct4`	Defines an extended IP access list using a name and enters extended named access list configuration mode.
Step 4	[*sequence-number*] **deny** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **Example:** `Router(config-ext-nacl)# deny ip any 172.20.1.1`	(Optional) Denies any packet that matches all of the conditions specified in the statement. • This statement will apply to nonfragmented packets and initial fragments.
Step 5	[*sequence-number*] **deny** *protocol source*[*source-wildcard*][*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **fragments**	(Optional) Denies any packet that matches all of the conditions specified in the statement • This statement will apply to noninitial fragments.

	Command or Action	Purpose
	Example: `Router(config-ext-nacl)# deny ip any 172.20.1.1 fragments`	
Step 6	[*sequence-number*] **permit** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **Example:** `Router(config-ext-nacl)# permit tcp any any`	Permits any packet that matches all of the conditions specified in the statement. • Every access list needs at least one **permit** statement. • If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.
Step 7	Repeat some combination of Steps 4 through 6 until you have specified the values on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 8	**end**
	Example:
	`Router(config-ext-nacl)# end`
		Ends configuration mode and returns the system to privileged EXEC mode.

Step 9	**show ip access-list** **Example:** `Router# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Configuration Examples for Refining an IP Access List

Example Resequencing Entries in an Access List

The following example shows an access list before and after resequencing. The starting value is 1, and increment value is 2. The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483647.

When an entry with no sequence number is entered, by default it has a sequence number of 10 more than the last entry in the access list.

```
Router# show access-list carls
Extended IP access list carls
    10 permit ip host 10.3.3.3 host 172.16.5.34
    20 permit icmp any any
    30 permit tcp any host 10.3.3.3
    40 permit ip host 10.4.4.4 any
    50 Dynamic test permit ip any any
    60 permit ip host 172.16.2.2 host 10.3.3.12
    70 permit ip host 10.3.3.3 any log
    80 permit tcp host 10.3.3.3 host 10.1.2.2
    90 permit ip host 10.3.3.3 any
    100 permit ip any any
Router(config)# ip access-list extended carls
Router(config)# ip access-list resequence carls 1 2
Router(config)# end
Router# show access-list carls
Extended IP access list carls
    1 permit ip host 10.3.3.3 host 172.16.5.34
    3 permit icmp any any
    5 permit tcp any host 10.3.3.3
    7 permit ip host 10.4.4.4 any
    9 Dynamic test permit ip any any
    11 permit ip host 172.16.2.2 host 10.3.3.12
    13 permit ip host 10.3.3.3 any log
    15 permit tcp host 10.3.3.3 host 10.1.2.2
    17 permit ip host 10.3.3.3 any
    19 permit ip any any
```

Example Adding an Entry with a Sequence Number

In the following example, an new entry (sequence number 15) is added to an access list:

```
Router# show ip access-list
Standard IP access list tryon
2 permit 10.4.4.2, wildcard bits 0.0.255.255
5 permit 10.0.0.44, wildcard bits 0.0.0.255
10 permit 10.0.0.1, wildcard bits 0.0.0.255
20 permit 10.0.0.2, wildcard bits 0.0.0.255
Router(config)# ip access-list standard tryon
Router(config-std-nacl)# 15 permit 10.5.5.5 0.0.0.255
Router# show ip access-list
Standard IP access list tryon
2 permit 10.4.0.0, wildcard bits 0.0.255.255
5 permit 10.0.0.0, wildcard bits 0.0.0.255
10 permit 10.0.0.0, wildcard bits 0.0.0.255
15 permit 10.5.5.0, wildcard bits 0.0.0.255
20 permit 10.0.0.0, wildcard bits 0.0.0.255
```

Example Adding an Entry with No Sequence Number

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
Router(config)# ip access-list standard resources
Router(config-std-nacl)# permit 10.1.1.1 0.0.0.255
Router(config-std-nacl)# permit 10.2.2.2 0.0.0.255
Router(config-std-nacl)# permit 10.3.3.3 0.0.0.255
Router# show access-list
Standard IP access list resources
10 permit 10.1.1.1, wildcard bits 0.0.0.255
20 permit 10.2.2.2, wildcard bits 0.0.0.255
30 permit 10.3.3.3, wildcard bits 0.0.0.255
Router(config)# ip access-list standard resources
Router(config-std-nacl)# permit 10.4.4.4 0.0.0.255
Router(config-std-nacl)# end
Router# show access-list
Standard IP access list resources
10 permit 10.1.1.1, wildcard bits 0.0.0.255
20 permit 10.2.2.2, wildcard bits 0.0.0.255
30 permit 10.3.3.3, wildcard bits 0.0.0.255
40 permit 10.4.4.4, wildcard bits 0.0.0.255
```

Example Time Ranges Applied to IP Access List Entries

The following example creates a time range called no-http, which extends from Monday to Friday from 8:00 a.m. to 6:00 p.m. That time range is applied to the **deny** statement, thereby denying HTTP traffic on Monday through Friday from 8:00 a.m. to 6:00 p.m.

The time range called udp-yes defines weekends from noon to 8:00 p.m. That time range is applied to the **permit** statement, thereby allowing UDP traffic on Saturday and Sunday from noon to 8:00 p.m. only. The access list containing both statements is applied to inbound packets on Ethernet interface 0.

```
time-range no-http
 periodic weekdays 8:00 to 18:00
!
time-range udp-yes
 periodic weekend 12:00 to 20:00
!
ip access-list extended strict
 deny tcp any any eq http time-range no-http
 permit udp any any time-range udp-yes
!
interface ethernet 0
 ip access-group strict in
```

Example Filtering IP Packet Fragments

In the following access list, the first statement will deny only noninitial fragments destined for host 172.16.1.1. The second statement will permit only the remaining nonfragmented and initial fragments that are destined for host 172.16.1.1 TCP port 80. The third statement will deny all other traffic. In order to block noninitial fragments for any TCP port, we must block noninitial fragments for all TCP ports, including port 80 for host

172.16.1.1. That is, non-initial fragments will not contain Layer 4 port information, so, in order to block such traffic for a given port, we have to block fragments for all ports.

```
access-list 101 deny ip any host 172.16.1.1 fragments
access-list 101 permit tcp any host 172.16.1.1 eq 80
access-list 101 deny ip any any
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Using the **time-range** command to establish time ranges	"Performing Basic System Management" chapter in the *Cisco IOS Network Management Configuration Guide*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Refining an IP Access List

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 11: Feature Information for Refining an IP Access List

Feature Name	Releases	Feature Configuration Information
Distributed Time-Based Access Lists	12.2(2)T	Before the introduction of this feature, time-based access lists were not supported on line cards for the Cisco 7500 series routers. If time-based access lists were configured, they behaved as normal access lists. If an interface on a line card were configured with a time-based access list, the packets switched into the interface were not distributed switched through the line card, but were forwarded to the Route Processor for processing. The Distributed Time-Based Access Lists feature allows packets destined for an interface configured with a time-based access list to be distributed switched through the line card.

CHAPTER 10

Displaying and Clearing IP Access List Data Using ACL Manageability

This module describes how to display the entries in an IP access list and the number of packets that have matched each entry. Users can get these statistics globally, or per interface and per incoming or outgoing traffic direction, by using the ACL Manageability feature. Viewing details of incoming and outgoing traffic patterns on various interfaces of a network device can help secure devices against attacks coming in on a particular interface. This module also describes how to clear counters so that the count of packets matching an access list entry will restart from zero.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About Displaying and Clearing IP Access List Data Using ACL Manageability

Benefits of ACL Manageability

Prior to Cisco IOS Release 12.4(6)T, the ACL infrastructure in Cisco IOS software maintained only global statistics for each ACE in an ACL. With this method, if an ACL is applied to multiple interfaces, the maintained ACE statistics are the sum of incoming and outgoing packet matches (hits) on all the interfaces on which that ACL is applied.

However, if ACE statistics are maintained per interface and per incoming or outgoing traffic direction, users can view specific details of incoming and outgoing traffic patterns and the effectiveness of ACEs on the various interfaces of a network device. This type of information is useful for securing devices against attacks coming in on a particular interface.

Support for Interface-Level ACL Statistics

With Cisco IOS Release 12.4(6)T, the ACL infrastructure in Cisco IOS software is now extended to support the maintenance, display, and clearing of ACE statistics per interface and per incoming or outgoing traffic direction for ACLs. This support is often referred to as "support for interface-level statistics."

Note If the same access-group ACL is also used by other features, the maintained interface statistics are not updated when a packet match is detected by the other features. In this case, the sum of all the interface level statistics that are maintained for an ACL may not add up to the global statistics for that ACL.

How to Display and Clear IP Access List Data

This section contains the following procedures for displaying IP access lists and the counts of packets that match (hit) each list, and for clearing IP access list counters.

Note Alternatively, if you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you. For more information, see the "IP Access List Logging" section of the "IP Access List Overview."

Displaying Global IP ACL Statistics

Perform this task to display all IP access lists on the router and counts of packets that have matched.

SUMMARY STEPS

1. **enable**
2. **show ip access-list** [*access-list-number* | *access-list-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip access-list** [*access-list-number* | *access-list-name*] **Example:** `Router# show ip access-list limited`	Displays IP access list information. • This example displays statistics for all interfaces that use the access list named "limited."

Displaying Interface-Level IP ACL Statistics

This section describes how to display IP ACE statistics per interface and per incoming or outgoing traffic direction for ACLs. This feature is known as ACL Manageability.

Note

- ACL Manageability supports:
  - Only nondistributed software switched platforms.
  - Standard and extended statically configured ACLs, and Threat Mitigation Service (TMS) dynamic ACEs.
- ACL Manageability does not support:
  - Reflexive and user-configured dynamic ACLs and dynamic ACE blocks, such as Firewall and Authentication Proxy.
  - Virtual-template and virtual-access interfaces.

>

SUMMARY STEPS

1. **enable**
2. **show ip access-list interface** *interface-name* [**in**| **out**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip access-list interface** *interface-name* [**in**| **out**] **Example:** `Router# show ip access-list interface FastEthernet 0/0 in`	Displays IP access list information. • This example displays statistics about traffic coming into the FastEthernet interface. • To display debugging information about ACL interface-level statistics, use the **debug ip access-list intstats** command.

Clearing the Access List Counters

The system counts how many packets match (hit) each line of an access list; the counters are displayed by the **show access-lists** EXEC command. Perform this task to clear the counters of an access list. You might do this if you are trying to determine a more recent count of packets that match an access list, starting from zero.

SUMMARY STEPS

1. **enable**
2. **clear ip access-list counters** {*access-list-number* | *access-list-name}*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear ip access-list counters** {*access-list-number* | *access-list-name}*	Clears IP access list counters.

	Command or Action	Purpose
	Example: `Router# clear access-list counters corpmark`	

Configuration Examples for Displaying and Clearing IP Access List Data Using ACL Manageability

Example Displaying Global IP ACL Statistics

The following example displays global statistics for ACL 150:

```
Router# show ip access-list 150

Extended IP access list 150
   10 permit ip host 10.1.1.1 any (3 matches)
   30 permit ip host 10.2.2.2 any (27 matches)
```

Example Displaying Input Statistics

The following example displays statistics on incoming packets gathered from the FastEthernet interface 0/1, associated with access list 150 (ACL number):

```
Router#
 show ip access-list interface FastEthernet 0/1 in
Extended IP access list 150 in
   10 permit ip host 10.1.1.1 any (3 matches)
   30 permit ip host 10.2.2.2 any (12 matches)
```

Example Displaying Output Statistics

The following example displays statistics on outgoing packets gathered from the FastEthernet interface 0/0:

```
Router#
 show ip access-list interface FastEthernet 0/0 out
Extended IP access list myacl out
    5 deny ip any 10.1.0.0 0.0.255.255
    10 permit udp any any eq snmp (6 matches)
```

Example Displaying Input and Output Statistics

Note If no direction is specified, any input and output ACLs applied to that interface are displayed.

The following example displays input and output statistics gathered from the FastEthernet interface 0/0:

```
Router#
 show ip access-list interface FastEthernet 0/0
Extended IP access list 150 in
   10 permit ip host 10.1.1.1 any
   30 permit ip host 10.2.2.2 any (15 matches)
Extended IP access list myacl out
    5 deny ip any 10.1.0.0 0.0.255.255
    10 permit udp any any eq snmp (6 matches)
```

Example Clearing Global and Interface Statistics for an IP Access List

The following example clears global and interface statistics for IP ACL 150:

```
Router#
 clear ip access-list counters 150
```

Example Clearing Global and Interface Statistics for All IP Access Lists

The following example clears global and interface statistics for all IP ACLs:

```
Router#
 clear ip access-list counters
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Security commands	*Cisco IOS Security Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Displaying IP Access List Information and Clearing Counters

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 12: Feature Information for Displaying and Clearing IP Access List Data Using ACL Manageability

Feature Name	Releases	Feature Information
ACL Manageability	12.4(6)T	The ACL Manageability feature enables users to display and clear Access Control Entry (ACE) statistics per interface and per incoming or outgoing traffic direction for access control lists (ACLs).

CHAPTER 11

Object Groups for ACLs

The Object Groups for ACLs feature lets you classify users, devices, or protocols into groups and apply those groups to access control lists (ACLs) to create access control policies for those groups. This feature lets you use object groups instead of individual IP addresses, protocols, and ports, which are used in conventional ACLs. This feature allows multiple access control entries (ACEs), but now you can use each ACE to allow an entire group of users to access a group of servers or services or to deny them from doing so.

In large networks, the number of ACLs can be large (hundreds of lines) and difficult to configure and manage, especially if the ACLs frequently change. Object group-based ACLs are smaller, more readable, and easier to configure and manage than conventional ACLs, simplifying static and dynamic ACL deployments for large user access environments on Cisco IOS routers.

Cisco IOS Firewall benefits from object groups, because they simplify policy creation (for example, group A has access to group A services).

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Object Groups for ACLs

- You can use object groups only in extended named and numbered ACLs.
- Object group-based ACLs support only IPv4 addresses.
- Object group-based ACLs support only Layer 3 interfaces (such as routed interfaces and VLAN interfaces). Object group-based ACLs do not support Layer 2 features such as VLAN ACLs (VACLs) or port ACLs (PACLs).
- Object group-based ACLs are not supported with IPsec.
- The highest number of object group-based ACEs supported in an ACL is 2048.

Information About Object Groups for ACLs

You can configure conventional ACEs and ACEs that refer to object groups in the same ACL.

You can use object group-based ACLs with quality of service (QoS) match criteria, Cisco IOS Firewall, Dynamic Host Configuration Protocol (DHCP), and any other features that use extended ACLs. In addition, you can use object group-based ACLs with multicast traffic.

When there are many inbound and outbound packets, using object group-based ACLs increases performance when compared to conventional ACLs. Also, in large configurations, this feature reduces the storage needed in NVRAM, because using object groups in ACEs means that you do not need to define an individual ACE for every address and protocol pairing.

Object Groups

An object group can contain a single object (such as a single IP address, network, or subnet) or multiple objects (such as a combination of multiple IP addresses, networks, or subnets).

A typical access control entry (ACE) allows a group of users to have access only to a specific group of servers. In an object group-based access control list (ACL), you can create a single ACE that uses an object group name instead of creating many ACEs (which requires each ACE to have a different IP address). A similar object group (such as a protocol port group) can be extended to provide access only to a set of applications for a user group. ACEs can have object groups for the source only, destination only, none, or both.

You can use object groups to separate the ownership of the components of an ACE. For example, each department in an organization controls its group membership, and the administrator owns the ACE itself to control which departments can contact one another.

You can use object groups in features that use Cisco Policy Language (CPL) class maps.

This feature supports two types of object groups for grouping ACL parameters: network object groups and service object groups. Use these object groups to group IP addresses, protocols, protocol services (ports), and Internet Control Message Protocol (ICMP) types.

Objects Allowed in Network Object Groups

A network object group is a group of any of the following objects:

- Any IP address—includes a range from 0.0.0.0 to 255.255.255.255 (This is specified using the **any** command.)
- Host IP addresses
- Hostnames
- Other network object groups
- Subnets

- Host IP addresses
- Network address of group members
- Nested object groups

Objects Allowed in Service Object Groups

A service object group is a group of any of the following objects:

- Source and destination protocol ports (such as Telnet or Simple Network Management Protocol [SNMP])
- Internet Control Message Protocol (ICMP) types (such as echo, echo-reply, or host-unreachable)
- Top-level protocols (such as Encapsulating Security Payload [ESP], TCP, or UDP)
- Other service object groups

ACLs Based on Object Groups

All features that use or reference conventional access control lists (ACLs) are compatible with object-group-based ACLs, and the feature interactions for conventional ACLs are the same with object-group-based ACLs. This feature extends the conventional ACLs to support object-group-based ACLs and also adds new keywords and the source and destination addresses and ports.

You can add, delete, or change objects in an object group membership list dynamically (without deleting and redefining the object group). Also, you can add, delete, or change objects in an object group membership list without redefining the ACL access control entry (ACE) that uses the object group. You can add objects to groups, delete them from groups, and then ensure that changes are correctly functioning within the object-group-based ACL without reapplying the ACL to the interface.

You can configure an object-group-based ACL multiple times with a source group only, a destination group only, or both source and destination groups.

You cannot delete an object group that is used within an ACL or a class-based policy language (CPL) policy.

How to Configure Object Groups for ACLs

To configure object groups for ACLs, you first create one or more object groups. These can be any combination of network object groups (groups that contain objects such as, host addresses and network addresses) or service object groups (which use operators such as **lt**, **eq**, **gt**, **neq**, and **range** with port numbers). Then, you create access control entries (ACEs) that apply a policy (such as **permit** or **deny**) to those object groups.

Creating a Network Object Group

A network object group that contains a single object (such as a single IP address, a hostname, another network object group, or a subnet) or nested objects (multiple network object groups can be defined in single network object group), is with a network object-group-based ACL to create access control policies for the objects.

Perform this task to create a network object group.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **object-group network** *object-group-name*
4. **description** *description-text*
5. **host** {*host-address* | *host-name*}
6. *network-address* {*/nn* | *network-mask*}
7. **group-object** *nested-object-group-name*
8. Repeat the steps until you have specified objects on which you want to base your object group.
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**object-group network** *object-group-name* **Example:** `Device(config)# object-group network my-network-object-group`	Defines the object group name and enters network object-group configuration mode.
Step 4	**description** *description-text* **Example:** `Device(config-network-group)# description test engineers`	(Optional) Specifies a description of the object group. • You can use up to 200 characters.

	Command or Action	Purpose

Step 5	**host** {*host-address* | *host-name*}
	
	Example:
	
	`Device(config-network-group)# host 209.165.200.237`
		(Optional) Specifies the IP address or name of a host.
		• If you specify a host address, you must use an IPv4 address.

Step 6	*network-address* {*/nn* | *network-mask*}
	
	Example:
	
	`Device(config-network-group)# 209.165.200.241 255.255.255.224`
		(Optional) Specifies a subnet object.
		• You must specify an IPv4 address for the network address. The default network mask is 255.255.255.255.

Step 7	**group-object** *nested-object-group-name* **Example:** `Device(config-network-group)# group-object my-nested-object-group`	(Optional) Specifies a nested (child) object group to be included in the current (parent) object group. • The type of child object group must match that of the parent (for example, if you are creating a network object group, you must specify another network object group as the child). • You can use duplicated objects in an object group only via nesting of group objects. For example, if object 1 is in both group A and group B, you can define a group C that includes both A and B. However, you cannot include a group object that causes the group hierarchy to become circular (for example, you cannot include group A in group B and then also include group B in group A). • You can use an unlimited number of levels of nested object groups (however, a maximum of two levels is recommended).
Step 8	Repeat the steps until you have specified objects on which you want to base your object group.	—

Step 9	**end**
	
	Example:
	
	`Device(config-network-group)# end`
		Exits network object-group configuration mode and returns to privileged EXEC mode.

Creating a Service Object Group

Use a service object group to specify TCP and/or UDP ports or port ranges. When the service object group is associated with an access control list (ACL), this service object-group-based ACL can control access to ports.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **object-group service** *object-group-name*
4. **description** *description-text*
5. *protocol*
6. {**tcp** | **udp** | **tcp-udp**} [**source** {{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*}] [{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*]
7. **icmp** *icmp-type*
8. **group-object** *nested-object-group-name*
9. Repeat the steps to specify the objects on which you want to base your object group.
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**object-group service** *object-group-name* **Example:** `Device(config)# object-group service my-service-object-group`	Defines an object group name and enters service object-group configuration mode.
Step 4	**description** *description-text* **Example:** `Device(config-service-group)# description test engineers`	(Optional) Specifies a description of the object group. • You can use up to 200 characters.
Step 5	*protocol* **Example:** `Device(config-service-group)# ahp`	(Optional) Specifies an IP protocol number or name.

	Command or Action	Purpose
Step 6	{**tcp** | **udp** | **tcp-udp**} [**source** {{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*}] [{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*] **Example:** `Device(config-service-group)# tcp-udp range 2000 2005`	(Optional) Specifies TCP, UDP, or both.
Step 7	**icmp** *icmp-type* **Example:** `Device(config-service-group)# icmp conversion-error`	(Optional) Specifies the decimal number or name of an Internet Control Message Protocol (ICMP) type.
Step 8	**group-object** *nested-object-group-name* **Example:** `Device(config-service-group)# group-object my-nested-object-group`	(Optional) Specifies a nested (child) object group to be included in the current (parent) object group. • The type of child object group must match that of the parent (for example, if you are creating a network object group, you must specify another network object group as the child). • You can use duplicated objects in an object group only via nesting of group objects. For example, if object 1 is in both group A and group B, you can define a group C that includes both A and B. However, you cannot include a group object that causes the group hierarchy to become circular (for example, you cannot include group A in group B and then also include group B in group A). • You can use an unlimited number of levels of nested object groups (however, a maximum of two levels is recommended).
Step 9	Repeat the steps to specify the objects on which you want to base your object group.	—

Step 10	**end**
	
	Example:
	
	`Device(config-service-group)# end`
		Exits service object-group configuration mode and returns to privileged EXEC mode.

Creating an Object-Group-Based ACL

When creating an object-group-based access control list (ACL), configure an ACL that references one or more object groups. As with conventional ACLs, you can associate the same access policy with one or more interfaces.

You can define multiple access control entries (ACEs) that reference object groups within the same object-group-based ACL. You can also reuse a specific object group in multiple ACEs.

Perform this task to create an object-group-based ACL.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. **remark** *remark*
5. **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
6. **remark** *remark*
7. **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
8. Repeat the steps to specify the fields and values on which you want to base your access list.
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *access-list-name* **Example:** `Device(config)# ip access-list extended nomarketing`	Defines an extended IP access list using a name and enters extended access-list configuration mode.
Step 4	**remark** *remark* **Example:** `Device(config-ext-nacl)# remark protect`	(Optional) Adds a comment about the configured access list entry. • A remark can precede or follow an access list entry. • In this example, the remark reminds the network administrator that the subsequent entry denies the Marketing network access to the interface.

	Command or Action	Purpose
	`server by denying access from the Marketing network`	

Step 5	**deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
	
	Example:
	`Device(config-ext-nacl)# deny ip 209.165.200.244 255.255.255.224 host 209.165.200.245 log`
	`Example based on object-group:`
	`Router(config)#object-group network my_network_object_group`
	`Router(config-network-group)#209.165.200.224 255.255.255.224`
	`Router(config-network-group)#exit`
	`Router(config)#object-group network my_other_network_object_group`
	`Router(config-network-group)#host 209.165.200.245`
	`Router(config-network-group)#exit`
	`Router(config)#ip access-list extended nomarketing`
	`Router(config-ext-nacl)#deny ip object-group my_network_object_group object-group my_other_network_object_group log`
		(Optional) Denies any packet that matches all conditions specified in the statement.
		• Optionally use the **object-group** *service-object-group-name* keyword and argument as a substitute for the *protocol.* argument
		• Optionally use the **object-group** *source-network-object-group-name* keyword and argument as a substitute for the *source source-wildcard.* arguments
		• Optionally use the **object-group** *destination-network-object-group-name* keyword and argument as a substitute for the *destination destination-wildcard.* arguments
		• If the *source-wildcard* or *destination-wildcard*is omitted, a wildcard mask of 0.0.0.0 is assumed, which matches all bits of the source or destination address, respectively.
		• Optionally use the **any** keyword as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.
		• Optionally use the **host** *source* keyword and argument to indicate a source and source wildcard of *source* 0.0.0.0 or the **host** *destination* keyword and argument to indicate a destination and destination wildcard of *destination* 0.0.0.0.
		• In this example, packets from all sources are denied access to the destination network 209.165.200.244. Logging messages about packets permitted or denied by the access list are sent to the facility configured by the **logging facility** command (for example, console, terminal, or syslog). That is, any packet that matches the access list will cause an informational logging message about the packet to be sent to the configured facility. The level of messages logged to the console is controlled by the **logging console** command.
		•

Step 6	**remark** *remark* **Example:** `Device(config-ext-nacl)# remark allow TCP from any source to any destination`	(Optional) Adds a comment about the configured access list entry. • A remark can precede or follow an access list entry.
Step 7	**permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log**	Permits any packet that matches all conditions specified in the statement. • Every access list needs at least one permit statement.

	Command or Action	Purpose
	| **log-input**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-ext-nacl)# permit tcp any any`	• Optionally use the **object-group** *service-object-group-name* keyword and argument as a substitute for the *protocol.* • Optionally use the **object-group** *source-network-object-group-name* keyword and argument as a substitute for the *source source-wildcard.* • Optionally use the **object-group** *destination-network-object-group-name* keyword and argument as a substitute for the *destination destination-wildcard.* • If *source-wildcard* or *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, which matches on all bits of the source or destination address, respectively. • Optionally use the **any**keyword as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255. • In this example, TCP packets are allowed from any source to any destination. • Use the **log-input** keyword to include input interface, source MAC address, or virtual circuit in the logging output.
Step 8	Repeat the steps to specify the fields and values on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 9	**end**
	
	Example:
	
	`Device(config-ext-nacl)# end`
		Exits extended access-list configuration mode and returns to privileged EXEC mode.

Applying an Object Group-Based ACL to an Interface

Use the **ip access-group** command to apply an object group-based ACL to an interface. An object group-based access control list (ACL) can be used to control traffic on the interface it is applied to.

Perform this task to apply an object group-based ACL to an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip access-group** {*access-list-name* | *access-list-number*} {**in** | **out**}
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface vlan 100`	Specifies the interface and enters interface configuration mode.

Step 4	**ip access-group** {*access-list-name* | *access-list-number*} {**in** | **out**}
	
	Example:
	
	`Device(config-if)# ip access-group my-ogacl-policy in`
		Applies the ACL to the interface and specifies whether to filter inbound or outbound packets.

Step 5	**end**
	
	Example:
	
	`Device(config-if)# end`
		Exits interface configuration mode and returns to privileged EXEC mode.

Verifying Object Groups for ACLs

SUMMARY STEPS

1. **enable**
2. **show object-group** [*object-group-name*]
3. **show ip access-list** [*access-list-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show object-group** [*object-group-name*] **Example:** `Device# show object-group my-object-group`	Displays the configuration in the named or numbered object group (or in all object groups if no name is entered).
Step 3	**show ip access-list** [*access-list-name*] **Example:** `Device# show ip access-list my-ogacl-policy`	Displays the contents of the named or numbered access list or object group-based ACL (or for all access lists and object group-based ACLs if no name is entered).

Configuration Examples for Object Groups for ACLs

Example: Creating a Network Object Group

The following example shows how to create a network object group named my-network-object-group, which contains two hosts and a subnet as objects:

```
Device> enable
Device# configure terminal
Device(config)# object-group network my-network-object-group
Device(config-network-group)# description test engineers
Device(config-network-group)# host 209.165.200.237
Device(config-network-group)# host 209.165.200.238

Device(config-network-group)# 209.165.200.241 255.255.255.224
Device(config-network-group)# end
```

The following example shows how to create a network object group named my-company-network, which contains two hosts, a subnet, and an existing object group (child) named my-nested-object-group as objects:

```
Device> enable
Device# configure terminal
Device(config)# object-group network my-company-network
Device(config-network-group)# host host1
Device(config-network-group)# host 209.165.200.242
Device(config-network-group)# 209.165.200.225 255.255.255.224
Device(config-network-group)# group-object my-nested-object-group
Device(config-network-group)# end
```

Example: Creating a Service Object Group

The following example shows how to create a service object group named my-service-object-group, which contains several ICMP, TCP, UDP, and TCP-UDP protocols and an existing object group named my-nested-object-group as objects:

```
Device> enable
Device# configure terminal
Device(config)# object-group service my-service-object-group
Device(config-service-group)# icmp echo
Device(config-service-group)# tcp smtp
Device(config-service-group)# tcp telnet
Device(config-service-group)# tcp source range 1 65535 telnet
Device(config-service-group)# tcp source 2000 ftp
Device(config-service-group)# udp domain
Device(config-service-group)# tcp-udp range 2000 2005
Device(config-service-group)# group-object my-nested-object-group
Device(config-service-group)# end
```

Example: Creating an Object Group-Based ACL

The following example shows how to create an object-group-based ACL that permits packets from the users in my-network-object-group if the protocol ports match the ports specified in my-service-object-group:

```
Device> enable
Device# configure terminal
Device(config)# ip access-list extended my-ogacl-policy
Device(config-ext-nacl)# permit object-group my-service-object-group object-group
my-network-object-group any
Device(config-ext-nacl)# deny tcp any any
Device(config-ext-nacl)# end
```

Example Applying an Object Group-Based ACL to an Interface

The following example shows how to apply an object group-based ACL to an interface. In this example, an object group-based ACL named my-ogacl-policy is applied to VLAN interface 100:

```
Device> enable
Device# configure terminal
Device(config)# interface vlan 100
Device(config-if)# ip access-group my-ogacl-policy in
Device(config-if)# end
```

Example: Verifying Object Groups for ACLs

The following example shows how to display all object groups:

```
Device# show object-group

Network object group auth-proxy-acl-deny-dest
 host 209.165.200.235
Service object group auth-proxy-acl-deny-services
 tcp eq www
 tcp eq 443
Network object group auth-proxy-acl-permit-dest
 209.165.200.226 255.255.255.224
 209.165.200.227 255.255.255.224
 209.165.200.228 255.255.255.224
 209.165.200.229 255.255.255.224
 209.165.200.246 255.255.255.224
 209.165.200.230 255.255.255.224
 209.165.200.231 255.255.255.224
 209.165.200.232 255.255.255.224
 209.165.200.233 255.255.255.224
 209.165.200.234 255.255.255.224
Service object group auth-proxy-acl-permit-services
 tcp eq www
 tcp eq 443
```

The following example shows how to display information about specific object-group-based ACLs:

```
Device# show ip access-list my-ogacl-policy

Extended IP access list my-ogacl-policy
10 permit object-group eng_service any any
```

Additional References for Object Groups for ACLs

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Security commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z

Related Topic	Document Title
ACL configuration guide	*Security Configuration Guide: Access Control Lists*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Object Groups for ACLs

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 13: Feature Information for Object Groups for ACLs

Feature Name	Releases	Feature Information
Object Groups for ACLs	12.4(20)T	The Object Groups for ACLs feature lets you classify users, devices, or protocols into groups and apply them to access control lists (ACLs) to create access control policies for those groups. This feature lets you use object groups instead of individual IP addresses, protocols, and ports, which are used in conventional ACLs. This feature allows multiple access control entries (ACEs), but now you can use each ACE to allow an entire group of users to access a group of servers or services or to deny them from doing so. The following commands were introduced or modified: **deny**, **ip access-group**, **ip access-list**, **object-group network**, **object-group service**, **permit**, **show ip access-list**, **show object-group**.

CHAPTER 12

Controlling Access to a Virtual Terminal Line

You can control who can access the virtual terminal lines (vtys) to a router by applying an access list to inbound vtys. You can also control the destinations that the vtys from a router can reach by applying an access list to outbound vtys.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Controlling Access to a Virtual Terminal Line

When you apply an access list to a vty (by using the **access-class** command), the access list must be a numbered access list, not a named access list.

Information About Controlling Access to a Virtual Terminal Line

Benefits of Controlling Access to a Virtual Terminal Line

By applying an access list to an inbound vty, you can control who can access the lines to a router. By applying an access list to an outbound vty, you can control the destinations that the lines from a router can reach.

How to Control Access to a Virtual Terminal Line

Controlling Inbound Access to a vty

Perform this task when you want to control access to a vty coming into the router by using an access list. Access lists are very flexible; this task illustrates one **access-list deny** command and one **access-list permit**command. You will decide how many of each command you should use and their order to achieve the restrictions you want.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* **deny** {*source* [*source-wildcard*] | **any**} [**log**]
4. **access-list** *access-list-number* **permit** {*source* [*source-wildcard*] | **any**}[**log**]
5. **line vty** *line-number* [*ending-line-number*]
6. **access-class** *access-list-number* **in** [**vrf-also**]
7. **exit**
8. Repeat Steps 5 and 6 for each line to set identical restrictions on all the vtys because a user can connect to any of them.
9. **end**
10. **show line** [*line-number* | **summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**access-list** *access-list-number* **deny** {*source* [*source-wildcard*] | **any**} [**log**] **Example:** `Router(config)# access-list 1 deny 172.16.7.34`	(Optional) Denies the specified source based on a source address and wildcard mask. • If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, host 172.16.7.34 is denied passing the access list.
Step 4	**access-list** *access-list-number* **permit** {*source* [*source-wildcard*] | **any**}[**log**] **Example:** `Router(config)# access-list 1 permit 172.16.0.0 0.0.255.255`	Permits the specified source based on a source address and wildcard mask. • If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, hosts on network 172.16.0.0 (other than the host denied in the prior step) pass the access list, meaning they can access the vtys identified in the **line** command.
Step 5	**line vty** *line-number* [*ending-line-number*] **Example:** `Router(config)# line vty 5 10`	Identifies a specific line for configuration and enters line configuration mode. • Entering the **line** command with the optional line type **vty** designates the line number as a relative line number. • You also can use the **line** command without specifying a line type. In this case, the line number is treated as an absolute line number.
Step 6	**access-class** *access-list-number* **in** [**vrf-also**] **Example:** `Router(config-line)# access-class 1 in vrf-also`	Restricts incoming connections between a particular vty (into a Cisco device) and the networking devices associated with addresses in the access list. • If you do not specify the **vrf-also** keyword, incoming Telnet connections from interfaces that are part of a VPN routing and forwarding (VRF) instance are rejected.

	Command or Action	Purpose
Step 7	**exit** **Example:** `Router(config-line)# exit`	Returns the user to the next highest configuration mode.
Step 8	Repeat Steps 5 and 6 for each line to set identical restrictions on all the vtys because a user can connect to any of them.	If you indicated the full range of vty lines in Step 5 with the **line** command, you do not need to repeat Steps 5 and 6.

Step 9	**end**
	
	Example:
	
	`Router(config-line)# end`
		Returns the user to privileged EXEC mode.

Step 10	**show line** [*line-number* | **summary**] **Example:** `Router# show line 5`	Displays parameters of a terminal line.

Controlling Outbound Access to a vty

Perform this task when you want to control access from a vty to a destination. Access lists are very flexible; this task illustrates one **access-list deny** command and one **access-list permit**command. You will decide how many of each command you should use and their order to achieve the restrictions you want.

When a standard access list is applied to a line with the **access-class out**command, the address specified in the access list is not a source address (as it is in an access list applied to an interface), but a destination address.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* **deny** {*destination* [*destination-wildcard*] | **any**} [**log**]
4. **access-list** *access-list-number* **permit** {*source* [*source-wildcard*] | **any**} [**log**]
5. **line vty** *line-number* [*ending-line-number*]
6. **access-class** *access-list-number* **out**
7. **exit**
8. Repeat Steps 5 and 6 for each line to set identical restrictions on all the vtys because a user can connect to any of them.
9. **end**
10. **show line** [*line-number* | **summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**access-list** *access-list-number* **deny** {*destination* [*destination-wildcard*] | **any**} [**log**] **Example:** `Router(config)# access-list 2 deny 172.16.7.34`	Denies line access to the specified destination based on a destination address and wildcard mask. • If the *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address. • Optionally use the keyword **any** as a substitute for the *destination destination-wildcard*to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, host 172.16.7.34 is denied passing the access list, meaning the line cannot connect to it.
Step 4	**access-list** *access-list-number* **permit** {*source* [*source-wildcard*] | **any**} [**log**] **Example:** `Router(config)# access-list 2 permit 172.16.0.0 0.0.255.255`	Permits the specified source based on a source address and wildcard mask. • If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*to specify the source and source wildcard of 0.0.0.0 255.255.255.255. • In this example, hosts on network 172.16.0.0 (other than the host denied in the prior step) pass the access list, meaning they can be connected to by the vtys identified in the **line** command.
Step 5	**line vty** *line-number* [*ending-line-number*] **Example:** `Router(config)# line vty 5 10`	Identifies a specific line for configuration and enter line configuration mode. • Entering the **line** command with the optional line type **vty** designates the line number as a relative line number. • You also can use the **line** command without specifying a line type. In this case, the line number is treated as an absolute line number.

	Command or Action	Purpose
Step 6	**access-class** *access-list-number* **out** **Example:** `Router(config-line)# access-class 2 out`	Restricts connections between a particular vty (into a Cisco device) out to the networking devices associated with addresses in the access list.
Step 7	**exit** **Example:** `Router(config-line)# exit`	Returns the user to the next highest configuration mode.
Step 8	Repeat Steps 5 and 6 for each line to set identical restrictions on all the vtys because a user can connect to any of them.	If you indicated the full range of vtys in Step 5 with the **line** command, you do not need to repeat Steps 5 and 6.

Step 9	**end**
	
	Example:
	
	`Router(config-line)# end`
		Returns the user to privileged EXEC mode.

Step 10	**show line** [*line-number* | **summary**] **Example:** `Router# show line 5`	Displays parameters of a terminal line.

Configuration Examples for Controlling Access to a Virtual Terminal Line

Example Controlling Inbound Access on vtys

The following example defines an access list that permits only hosts on network 172.19.5.0 to connect to the virtual terminal lines 1 through 5 on the router. Because the **vty** keyword is omitted from the **line** command, the line numbers 1 through 5 are absolute line numbers.

```
access-list 12 permit 172.19.5.0 0.0.0.255
line 1 5
 access-class 12 in
```

Example Controlling Outbound Access on vtys

The following example defines an access list that denies connections to networks other than network 171.20.0.0 on terminal lines 1 through 5. Because the **vty** keyword is omitted from the **line** command, the line numbers 1 through 5 are absolute line numbers.

```
access-list 10 permit 172.20.0.0 0.0.255.255
line 1 5
 access-class 10 out
```

Where to Go Next

You can further secure a vty by configuring a password with the **password** line configuration command. See the **password** (line configuration) command in the *Cisco IOS Security Command Reference*.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Configuring a password on a line	*Cisco IOS Security Command Reference*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Controlling Access to a Virtual Terminal Line

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 14: Feature Information for Controlling Access to a Virtual Terminal Line

Feature Name	Releases	Feature Configuration Information
Controlling Access to a Virtual Terminal Line	12.0(32)S4	You can control who can access the virtual terminal lines (vtys) to a router by applying an access list to inbound vtys. You can also control the destinations that the vtys from a router can reach by applying an access list to outbound vtys.

CHAPTER 13

Access List-Based RBSCP

The Access List-Based Rate-Based Satellite Control Protocol (RBSCP) feature allows you to selectively apply the TCP ACK splitting feature of RBSCP to any outgoing interface. The result is reduced effect of long latencies over a satellite link. Access List-Based RBSCP has no tunneling or queueing overhead that is associated with RBSCP tunnels. Additional benefits include more interoperability with other Cisco IOS features (such as TCP/IP header compresssion, DMVPN, and QoS) because the TCP and Stream Control Transmission Protocol (SCTP) packets are no longer encapsulated with an RBSCP/IP header. This feature works on process switched forwarding, fast switching, or Cisco Express Forwarding (CEF).

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Access List-Based RBSCP

This document assumes that you already understand how to configure an IP access list and have one configured.

Restrictions for Access List-Based RBSCP

Caution Plan your network carefully so that no more than one Cisco IOS router in a given routing path has the Access List-Based RBSCP feature enabled. You do not want to recursively ACK split traffic.

- The Access List-Based RBSCP feature will process only IPv4 packets, not IPv6 packets.
- The feature will process only standalone TCP packets. Encapsulated (encrypted or tunneled) TCP packets will be left unprocessed.
- This feature is available only on non-distributed platforms.

Information About Access List-Based RBSCP

Benefits of Access List-Based RBSCP

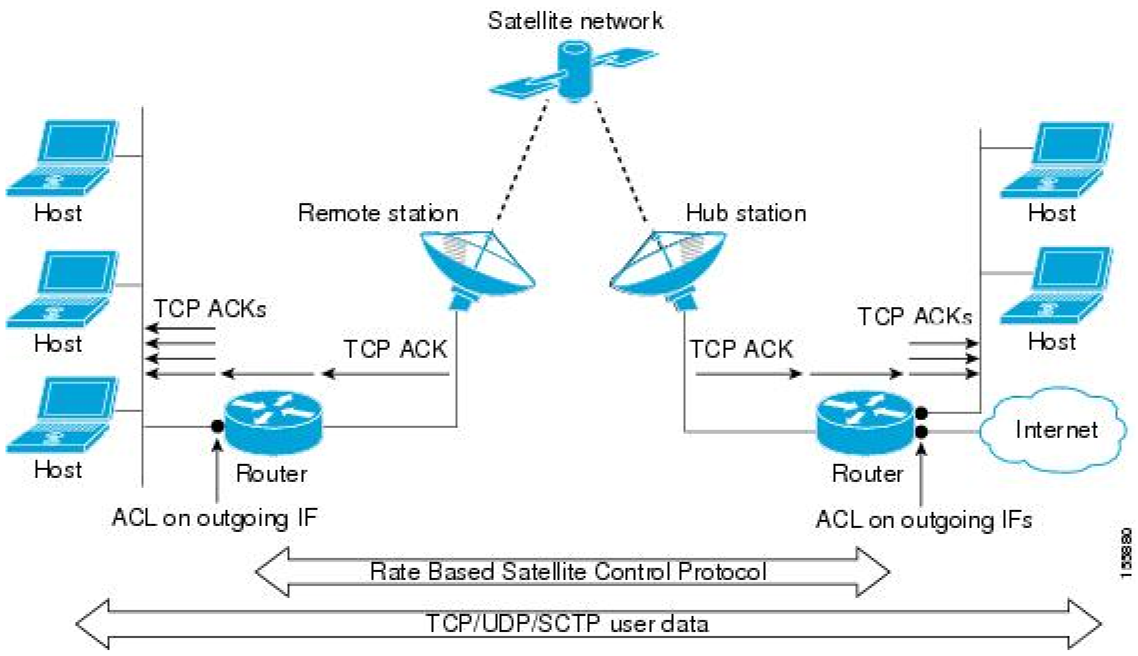
The Access List-Based Rate-Based Satellite Control Protocol (RBSCP) feature provides the following benefits:

- It allows you to selectively apply the TCP ACK splitting feature of RBSCP to any outgoing interface. TCP ACK splitting is a benefit because it reduces the effect of long latencies characteristic of satellite links. Applying this feature selectively by using an access list is a benefit because you control which packets are subject to TCP ACK splitting.
- It has no tunneling or queueing overhead that is associated with RBSCP tunnels.
- It provides more interoperability with other Cisco IOS features (such as TCP/IP header compresssion, DMVPN, and QoS) because the TCP and Stream Control Transmission Protocol (SCTP) packets are no longer encapsulated with an RBSCP/IP header.
- This feature works on process switched forwarding, fast switching, or CEF.
- It preserves the internet end-to-end principle.

Rate-Based Satellite Control Protocol

Rate-Based Satellite Control Protocol (RBSCP) was designed for wireless or long-distance delay links with high error rates, such as satellite links. RBSCP can improve the performance of certain IP protocols, such as TCP and IP Security (IPsec), over satellite links without breaking the end-to-end model. For instructions on how to implement RBSCP over a tunnel, see the “Implementing Tunnels” chapter of the *Interface and Hardware Component Configuration Guide* .

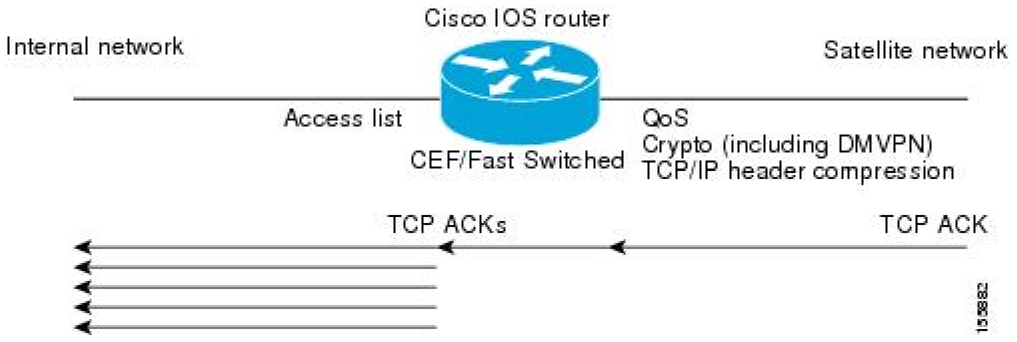
The TCP ACK splitting capability of RBSCP can be implemented without a tunnel, by using an IP access list, as shown in the figure below. The TCP ACK splitting occurs at the outgoing interface between the router and the internal network or Internet. It does not occur over the link to the satellite.



TCP ACK Splitting

TCP ACK splitting is a software technique to improve performance for clear-text TCP traffic using acknowledgment (ACK) splitting, in which a number of additional TCP ACKs are generated for each TCP ACK received. TCP ACK splitting causes TCP to open the congestion window more quickly than usual, thus decreasing the effect of long latencies. TCP will generally open the congestion window by one maximum transmission unit (MTU) for each TCP ACK received. Opening the congestion window results in increased bandwidth becoming available. Configure this feature only when the satellite link is not using all the available bandwidth. Encrypted traffic cannot use TCP ACK splitting.

The *size* argument in the **ip rbscp ack-split**command determines how many TCP ACKs are generated from the incoming TCP ACK, as shown in the figure below.



If n ACKs are configured and M is the cumulative ACK point of the original TCP ACK, the resulting TCP ACKs exiting the router will have the following cumulative ACK points:

M-n+1, M-n+2, M-n+3,...M

For example, if the *size* argument is set to 5, and the access list permits a TCP ACK with a cumulative ACK acknowledging bytes to 1000, then the resulting TCP ACKs exiting the router will have the following cumulative ACK points:

TCP ACK (996) (1000-5+1)

TCP ACK (997) (1000-5+2)

TCP ACK (998) (1000-5+3)

TCP ACK (999) (1000-5+4)

TCP ACK (1000) (1000-5+5)

Access List-Based RBSCP Functionality

The Access List-Based RBSCP feature will accept a numbered or named, standard or extended IP access list. The access list controls which packets are subject to TCP ACK splitting. That is, the feature is applied to packets that a **permit** statement allows; the feature is not applied to packets that a **deny** statement filters.

An instance of this feature consists of an access list and an ACK split value. An ACK split value of 0 or 1 indicates that this feature is disabled (that is, no ACK split will be done). The ACK split value range is 0 through 32.

An interface can use only one instance of this feature at a time. Each instance of this feature can be used on multiple interfaces.

If you configure this feature but it refers to a nonexistent access list, this is interpreted as having an access list that denies all traffic from being processed by the access list-based RBSCP feature, so the feature is essentially disabled and the traffic goes through the normal switching path.

If both an RBSCP tunnel and an instance of the Access List-Based RBSCP feature are enabled along a routing or switching path, the TCP ACKs detunneled from the RBSCP tunnel will be ACK split according to the tunnel configuration and the Access List-Based RBSCP split parameters on the outgoing interface are effectively disabled.

How to Configure Access List-Based RBSCP

Use RBSCP Selectively by Applying an Access List

This task illustrates how to apply the feature to an interface, and presumes that an access list is already configured. Perform this task by applying the access list on the router interface that is facing the internal network, not the satellite network.

Tip The feature will try to process all the TCP flows as filtered by the access list. Try to make the access list applied to RBSCP as precise as possible to avoid unnecessary processing.

Caution Plan your network carefully so that no more than one Cisco IOS router in a given routing path has this feature enabled. You do not want to recursively ACK split traffic.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip rbscp ack-split** *size* {*access-list-name* | *access-list-number*} **out**
5. Although it is not required, you should repeat this task on the router that is on the other side of the satellite, on the outgoing interface facing the network, not the satellite. Use a different access list.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface ethernet 1`	Specifies an interface. • Specify an interface that is facing your internal network, opposite the satellite network.
Step 4	**ip rbscp ack-split** *size* {*access-list-name* | *access-list-number*} **out** **Example:** `Router(config-if)# ip rbscp ack-split 6 101 out`	Configures RBSCP on the outgoing interface for packets that are permitted by the specified access list. • The ACK split *size* determines the number of ACKs to send for every ACK received. An ACK split value of 0 or 1 indicates that this feature is disabled (that is, no ACK split will be done). The range is 0 through 32. See "TCP ACK Splitting". • In this example, access list 101 determines which packets are subject to TCP ACK splitting.
Step 5	Although it is not required, you should repeat this task on the router that is on the other side of the satellite, on the outgoing interface facing the network, not the satellite. Use a different access list.	--

Configuration Examples for Access List-Based RBSCP

Example Access List-Based RBSCP

In the following example, access list 101 performs TCP ACK splitting on packets going out FastEthernet interface 1/1 from a source at 1.1.1.1 to a destination at 3.3.3.1:

```
!
version 12.4
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname IOSACL-72b
!
boot-start-marker
boot-end-marker
!
enable password lab
!
no aaa new-model
!
resource policy
!
ip cef
!
interface Ethernet0/0
 no ip address
 shutdown
 duplex auto
 no cdp enable
!
interface GigabitEthernet0/0
 no ip address
 shutdown
 duplex full
 speed 1000
 media-type gbic
 negotiation auto
 no cdp enable
!
interface FastEthernet1/0
 ip address 1.1.1.2 255.255.255.0
 duplex half
 no cdp enable
!
interface FastEthernet1/1
 ip address 2.2.2.2 255.255.255.0
 ip rbscp ack-split 4 101 out
 duplex half
 no cdp enable
!
interface FastEthernet2/0
 no ip address
 shutdown
 duplex half
 no cdp enable
!
interface Serial3/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/1
```

```
 no ip address
 shutdown
 serial restart-delay 0
 no cdp enable
!
interface Serial3/2
 no ip address
 shutdown
 serial restart-delay 0
 no cdp enable
!
interface Serial3/3
 no ip address
 shutdown
 serial restart-delay 0
 no cdp enable
!
interface FastEthernet4/0
 no ip address
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
interface FastEthernet4/1
 no ip address
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
router eigrp 100
 network 1.0.0.0
 network 2.0.0.0
 auto-summary
!
no ip http server
no ip http secure-server
!
logging alarm informational
access-list 101 permit tcp host 1.1.1.1 host 3.3.3.1
dialer-list 1 protocol ip permit
!
control-plane
!
gatekeeper
 shutdown
!
!
line con 0
 exec-timeout 0 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Related Topic	Document Title
IP access list commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*
RBSCP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Interface and Hardware Component Command Reference*
Configuring Rate-Based Satellite Control Protocol (RBSCP)	"Implementing Tunnels" chapter in the *Cisco IOS Interface and Hardware Component Configuration Guide*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Access List-Based RBSCP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 15: Feature Information for Access List-Based RBSCP

Feature Name	Releases	Feature Information
Access List-Based RBSCP	12.4(9)T	The Access List-Based Rate-Based Satellite Control Protocol feature allows you to selectively apply the TCP ACK splitting sub-feature of RBSCP to any outgoing interface. This feature has no tunneling or queueing overhead that is associated with RBSCP tunnels. The following commands are introduced or modified by this feature: **debug ip rbscp**, **debug ip rbscp ack-split**, **ip rbscp ack-split**.

CHAPTER 14

ACL IP Options Selective Drop

The ACL IP Options Selective Drop feature allows Cisco routers to filter packets containing IP options or to mitigate the effects of IP options on a router or downstream routers by dropping these packets or ignoring the processing of the IP options.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for ACL IP Options Selective Drop

- Resource Reservation Protocol (RSVP) (Multiprotocol Label Switching traffic engineering [MPLS TE]), Internet Group Management Protocol Version 2 (IGMPv2), and other protocols that use IP options packets may not function in drop or ignore modes.
- On the Cisco 10720 Internet router, the **ip option ignore**command is not supported. Only drop mode (the **ip option drop**command) is supported.

- The **ip option ignore** command (ignore mode) is supported only on the Cisco 12000 series router.

Information About ACL IP Options Selective Drop

Using ACL IP Options Selective Drop

The ACL IP Options Selective Drop feature allows a router to filter IP options packets, thereby mitigating the effects of these packets on a router and downstream routers, and perform the following actions:

- Drop all IP options packets that it receives and prevent options from going deeper into the network.
- Ignore IP options packets destined for the router and treat them as if they had no IP options.

For many users, dropping the packets is the best solution. However, in environments in which some IP options may be legitimate, reducing the load that the packets present on the routers is sufficient. Therefore, users may prefer to skip options processing on the router and forward the packet as though it were pure IP.

Benefits of Using ACL IP Options Selective Drop

- Drop mode filters packets from the network and relieves downstream routers and hosts of the load from options packets.
- Drop mode minimizes loads to the Route Processor (RP) for options that require RP processing on distributed systems. Previously, the packets were always routed to or processed by the RP CPU. Now, the ignore and drop forms prevent the packets from impacting the RP performance.

How to Configure ACL IP Options Selective Drop

Configuring ACL IP Options Selective Drop

This section describes how to configure the ACL IP Options Selective Drop feature.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip options** {**drop** | **ignore**}
4. **exit**
5. **show ip traffic**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**ip options** {**drop** | **ignore**}
	
	Example:
	
	`Router(config)# ip options drop`
		Drops or ignores IP options packets that are sent to the router.
		Note On the Cisco 10720 Internet router, the **ip option ignore**command is not supported. Only drop mode (the **ip option drop**command) is supported.

Step 4	**exit** **Example:** `Router(config)# exit`	Returns to privileged EXEC mode.
Step 5	**show ip traffic** **Example:** `Router# show ip traffic`	(Optional) Displays statistics about IP traffic.

What to Do Next

If you are running Cisco IOS Release 12.3(4)T or a later release, you can also use the ACL Support for Filtering IP Options feature to filter packets based on whether the packet contains specific IP options. For more information, refer to the document "Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values".

Configuration Example for ACL IP Options Selective Drop

Example Configuring ACL IP Options Selective Drop

The following example shows how to configure the router (and downstream routers) to drop all options packets that enter the network:

```
Router(config)# ip options drop
% Warning:RSVP and other protocols that use IP Options packets may not function in drop or
 ignore modes.
end
```

Example Verifying ACL IP Options Selective Drop

The following sample output is displayed after 15,000 options packets are sent using the **ip options drop** command. Note that the “forced drop” counter increases.

```
Router# show ip traffic
IP statistics:
  Rcvd:  15000 total, 0 local destination
         0 format errors, 0 checksum errors, 0 bad hop count
         0 unknown protocol, 0 not a gateway
         0 security failures, 0 bad options, 15000 with options
  Opts:  0 end, 0 nop, 0 basic security, 0 loose source route
         0 timestamp, 0 extended security, 0 record route
         0 stream ID, 0 strict source route, 0 alert, 0 cipso
         0 other
  Frags: 0 reassembled, 0 timeouts, 0 couldn't reassemble
         0 fragmented, 0 couldn't fragment
  Bcast: 0 received, 0 sent
  Mcast: 0 received, 0 sent
  Sent:  0 generated, 0 forwarded
  Drop:  0 encapsulation failed, 0 unresolved, 0 no adjacency
         0 no route, 0 unicast RPF, 15000 forced drop
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Configuring IP access lists	"Creating an IP Access List and Applying It to an Interface"
Using access lists for filtering IP options	"Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values"

Standards

Standards	Title
None	--

MIBs

MIBs	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for ACL IP Options Selective Drop

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 16: Feature Information for ACL IP Options Selective Drop

Feature Name	Releases	Feature Information
ACL IP Options Selective Drop	12.0(22)S 12.3(4)T 12.2(25)S 12.2(27)SBC 12.0(32)S 12.3(19)	The ACL IP Options Selective Drop feature allows Cisco routers to filter packets containing IP options or to mitigate the effects of IP options on a router or downstream routers by dropping these packets or ignoring the processing of the IP options. The following commands were introduced or modified: **ip options**.

CHAPTER 15

ACL Authentication of Incoming rsh and rcp Requests

This document describes the ACL Authentication of Incoming RSH and RCP Requests feature in Cisco IOS Release 12.2(8)T.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Overview of ACL Authentication of Incoming rsh and rcp Requests

To enable the Cisco IOS software to receive incoming remote shell (rsh) protocol and remote copy (rcp) protocol requests, customers must configure an authentication database to control access to the router. This configuration is accomplished by using the **ip rcmd remote-host** command.

Currently, when using this command, customers must specify the local user, the remote host, and the remote user in the database authentication configuration. For users who can execute commands to the router from multiple hosts, multiple database authentication configuration entries must be used, one for each host, as shown below.

```
ip rcmd remote-host local-user1 remote-host1 remote-user1
ip rcmd remote-host local-user1 remote-host2 remote-user1
ip rcmd remote-host local-user1 remote-host3 remote-user1
ip rcmd remote-host local-user1 remote-host4 remote-user1
```

This feature allows customers to specify an access list for a given user. The access list identifies the hosts to which the user has access. A new argument, *access-list*, has been added that can be used with this command to specify the access list, as shown below.

```
ip rcmd remote-host local-user1 access-list remote-user1
```

To allow a user access to the hosts identified in the access list, first define the access list. If the access list is not already defined, access to the host will be denied. For information about defining an access list, refer to the *Cisco IOS Security Configuration Guide* .

Supported Platforms

- Cisco 805
- Cisco 806
- Cisco 828
- Cisco 1400 series
- Cisco 1600 series
- Cisco 1710
- Cisco 1720
- Cisco 1721
- Cisco 1750
- Cisco 1751
- Cisco 2420
- Cisco 3620
- Cisco 3631
- Cisco 3640
- Cisco 3660
- Cisco 3725
- Cisco 3745
- Cisco 2500 series
- Cisco 2600 series
- Cisco 7100 series
- Cisco 7200 series

- Cisco 7500 series
- Cisco uBR7200 series
- Cisco Voice Gateway 200
- URM (Universal Route Module)

Additional References for Firewall TCP SYN Cookie

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Security commands	• Security Command Reference: Commands A to C • Security Command Reference: Commands D to L • Security Command Reference: Commands M to R • Security Command Reference: Commands S to Z

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for ACL Authentication of Incoming rsh and rcp Requests

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 17: Feature Information for ACL Authentication of Incoming rsh and rcp Requests

Feature Name	Releases	Feature Information
ACL Authentication of Incoming rsh and rcp Requests	12.2(8)T	This document describes the ACL Authentication of Incoming RSH and RCP Requests feature in Cisco IOS Release 12.2(8)T The following commands were introduced or modified: **ip rcmd remote-host**.

CHAPTER 16

Configuring Lock-and-Key Security (Dynamic Access Lists)

Feature History

Release	Modification
Cisco IOS	For information about feature support in Cisco IOS software, use Cisco Feature Navigator.

This chapter describes how to configure lock-and-key security at your router. Lock-and-key is a traffic filtering security feature available for the IP protocol.

For a complete description of lock-and-key commands, refer to the *Cisco IOS Security Command Reference* . To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release.

Prerequisites for Configuring Lock-and-Key

Lock-and-key uses IP extended access lists. You must have a solid understanding of how access lists are used to filter traffic, before you attempt to configure lock-and-key. Access lists are described in the chapter "Access Control Lists: Overview and Guidelines."

Lock-and-key employs user authentication and authorization as implemented in Cisco's authentication, authorization, and accounting (AAA) paradigm. You must understand how to configure AAA user authentication and authorization before you configure lock-and-key. User authentication and authorization is explained in the "Authentication, Authorization, and Accounting (AAA)" part of this document.

Lock-and-key uses the **autocommand** command, which you should understand. This command is described in the *Cisco IOSTerminal Services Command Reference.*

Information About Configuring Lock-and-Key Security (Dynamic Access Lists)

About Lock-and-Key

Lock-and-key is a traffic filtering security feature that dynamically filters IP protocol traffic. Lock-and-key is configured using IP dynamic extended access lists. Lock-and-key can be used in conjunction with other standard access lists and static extended access lists.

When lock-and-key is configured, designated users whose IP traffic is normally blocked at a router can gain temporary access through the router. When triggered, lock-and-key reconfigures the interface's existing IP access list to permit designated users to reach their designated host(s). Afterwards, lock-and-key reconfigures the interface back to its original state.

For a user to gain access to a host through a router with lock-and-key configured, the user must first open a Telnet session to the router. When a user initiates a standard Telnet session to the router, lock-and-key automatically attempts to authenticate the user. If the user is authenticated, they will then gain temporary access through the router and be able to reach their destination host.

Benefits of Lock-and-Key

Lock-and-key provides the same benefits as standard and static extended access lists (these benefits are discussed in the chapter "Access Control Lists: Overview and Guidelines"). However, lock-and-key also has the following security benefits over standard and static extended access lists:

- Lock-and-key uses a challenge mechanism to authenticate individual users.
- Lock-and-key provides simpler management in large internetworks.
- In many cases, lock-and-key reduces the amount of router processing required for access lists.
- Lock-and-key reduces the opportunity for network break-ins by network hackers.

With lock-and-key, you can specify which users are permitted access to which source and destination hosts. These users must pass a user authentication process before they are permitted access to their designated hosts. Lock-and-key creates dynamic user access through a firewall, without compromising other configured security restrictions.

When to Use Lock-and-Key

Two examples of when you might use lock-and-key follow:

- When you want a specific remote user (or group of remote users) to be able to access a host within your network, connecting from their remote hosts via the Internet. Lock-and-key authenticates the user, then permits limited access through your firewall router for the individual's host or subnet, for a finite period of time.
- When you want a subset of hosts on a local network to access a host on a remote network protected by a firewall. With lock-and-key, you can enable access to the remote host only for the desired set of local user's hosts. Lock-and-key require the users to authenticate through a TACACS+ server, or other security server, before allowing their hosts to access the remote hosts.

How Lock-and-Key Works

The following process describes the lock-and-key access operation:

1. A user opens a Telnet session to a border (firewall) router configured for lock-and-key. The user connects via the virtual terminal port on the router.
2. The Cisco IOS software receives the Telnet packet, opens a Telnet session, prompts for a password, and performs a user authentication process. The user must pass authentication before access through the router is allowed. The authentication process can be done by the router or by a central access security server such as a TACACS+ or RADIUS server.
3. When the user passes authentication, they are logged out of the Telnet session, and the software creates a temporary entry in the dynamic access list. (Per your configuration, this temporary entry can limit the range of networks to which the user is given temporary access.)
4. The user exchanges data through the firewall.
5. The software deletes the temporary access list entry when a configured timeout is reached, or when the system administrator manually clears it. The configured timeout can either be an idle timeout or an absolute timeout.

Note The temporary access list entry is not automatically deleted when the user terminates a session. The temporary access list entry remains until a configured timeout is reached or until it is cleared by the system administrator.

Compatibility with Releases Before Cisco IOS Release 11.1

Enhancements to the **access-list** command are used for lock-and-key. These enhancements are backward compatible--if you migrate from a release before Cisco IOS Release 11.1 to a newer release, your access lists will be automatically converted to reflect the enhancements. However, if you try to use lock-and-key with a release before Cisco IOS Release 11.1, you might encounter problems as described in the following caution paragraph:

Caution Cisco IOS releases before Release 11.1 are not upwardly compatible with the lock-and-key access list enhancements. Therefore, if you save an access list with software older than Release 11.1, and then use this software, the resulting access list will not be interpreted correctly. This could cause you severe security problems. You must save your old configuration files with Cisco IOS Release 11.1 or later software before booting an image with these files.

Risk of Spoofing with Lock-and-Key

Caution Lock-and-key access allows an external event (a Telnet session) to place an opening in the firewall. While this opening exists, the router is susceptible to source address spoofing.

When lock-and-key is triggered, it creates a dynamic opening in the firewall by temporarily reconfiguring an interface to allow user access. While this opening exists, another host might spoof the authenticated user's address to gain access behind the firewall. Lock-and-key does not cause the address spoofing problem; the problem is only identified here as a concern to the user. Spoofing is a problem inherent to all access lists, and lock-and-key does not specifically address this problem.

To prevent spoofing, configure encryption so that traffic from the remote host is encrypted at a secured remote router, and decrypted locally at the router interface providing lock-and-key. You want to ensure that all traffic using lock-and-key will be encrypted when entering the router; this way no hackers can spoof the source address, because they will be unable to duplicate the encryption or to be authenticated as is a required part of the encryption setup process.

Router Performance Impacts with Lock-and-Key

When lock-and-key is configured, router performance can be affected in the following ways:

- When lock-and-key is triggered, the dynamic access list forces an access list rebuild on the silicon switching engine (SSE). This causes the SSE switching path to slow down momentarily.
- Dynamic access lists require the idle timeout facility (even if the timeout is left to default) and therefore cannot be SSE switched. These entries must be handled in the protocol fast-switching path.
- When remote users trigger lock-and-key at a border router, additional access list entries are created on the border router interface. The interface's access list will grow and shrink dynamically. Entries are dynamically removed from the list after either the idle-timeout or max-timeout period expires. Large access lists can degrade packet switching performance, so if you notice performance problems, you should look at the border router configuration to see if you should remove temporary access list entries generated by lock-and-key.

Maintaining Lock-and-Key

When lock-and-key is in use, dynamic access lists will dynamically grow and shrink as entries are added and deleted. You need to make sure that entries are being deleted in a timely way, because while entries exist, the risk of a spoofing attack is present. Also, the more entries there are, the bigger the router performance impact will be.

If you do not have an idle or absolute timeout configured, entries will remain in the dynamic access list until you manually remove them. If this is the case, make sure that you are extremely vigilant about removing entries.

Dynamic Access Lists

Use the following guidelines for configuring dynamic access lists:

- Do not create more than one dynamic access list for any one access list. The software only refers to the first dynamic access list defined.
- Do not assign the same *dynamic-name* to another access list. Doing so instructs the software to reuse the existing list. All named entries must be globally unique within the configuration.
- Assign attributes to the dynamic access list in the same way you assign attributes for a static access list. The temporary access list entries inherit the attributes assigned to this list.
- Configure Telnet as the protocol so that users must open a Telnet session into the router to be authenticated before they can gain access through the router.
- Either define an idle timeout now with the **timeout** keyword in the **access-enable** command in the **autocommand** command, or define an absolute timeout value later with the **access-list** command. You must define either an idle timeout or an absolute timeout--otherwise, the temporary access list entry will remain configured indefinitely on the interface (even after the user has terminated their session) until the entry is removed manually by an administrator. (You could configure both idle and absolute timeouts if you wish.)
- If you configure an idle timeout, the idle timeout value should be equal to the WAN idle timeout value.
- If you configure both idle and absolute timeouts, the idle timeout value must be less than the absolute timeout value.
- If you realize that a job will run past the ACL’s absolute timer, use the **access-list dynamic-extend** command to extend the absolute timer of the dynamic ACL by six minutes. This command allows you to open a new Telnet session into the router to re-authentication yourself using lock-and-key.
- The only values replaced in the temporary entry are the source or destination address, depending whether the access list was in the input access list or output access list. All other attributes, such as port, are inherited from the main dynamic access list.
- Each addition to the dynamic list is always put at the beginning of the dynamic list. You cannot specify the order of temporary access list entries.
- Temporary access list entries are never written to NVRAM.
- To manually clear or to display dynamic access lists, refer to the section “Maintaining Lock-and-Key” later in this chapter.

Lock-and-Key Authentication

There are three possible methods to configure an authentication query process. These three methods are described in this section.

Note Cisco recommends that you use the TACACS+ server for your authentication query process. TACACS+ provides authentication, authorization, and accounting services. It also provides protocol support, protocol specification, and a centralized security database. Using a TACACS+ server is described in the next section, "Method 1--Configuring a Security Server."

Use a network access security server such as TACACS+ server. This method requires additional configuration steps on the TACACS+ server but allows for stricter authentication queries and more sophisticated tracking capabilities.

```
Router(config-line)# login tacacs
```

Use the **username** command. This method is more effective because authentication is determined on a user basis.

```
Router(config)# username

name
 {nopassword
 |
password
 {
mutual-password
 |
encryption-type

encryption-password
}}
```

Use the **password** and **login** commands. This method is less effective because the password is configured for the port, not for the user. Therefore, any user who knows the password can authenticate successfully.

```
R
outer(config-line)# password

password
Router(config-line)# login local
```

The autocommand Command

The **autocommand** command configures the system to automatically execute a specified privileged EXEC command when a user connects to a particular line. Use the following guidelines for configuring the **autocommand** command:

- If you use a TACACS+ server to authenticate the user, you should configure the **autocommand** command on the TACACS+ server as a per-user autocommand. If you use local authentication, use the **autocommand** command on the line.
- Configure all virtual terminal (VTY) ports with the same **autocommand** command. Omitting an **autocommand** command on a VTY port allows a random host to gain privileged EXEC mode access to the router and does not create a temporary access list entry in the dynamic access list.
- If you do not define an idle timeout with the **autocommand access-enable** command, you must define an absolute timeout with the **access-list** command. You must define either an idle timeout or an absolute timeout--otherwise, the temporary access list entry will remain configured indefinitely on the interface (even after the user has terminated the session) until the entry is removed manually by an administrator. (You could configure both idle and absolute timeouts if you wish.)

- If you configure both idle and absolute timeouts, the absolute timeout value must be greater than the idle timeout value.

How to Configure Lock-and-Key Security (Dynamic Access Lists)

Configuring Lock-and-Key

To configure lock-and-key, use the following commands beginning in global configuration mode. While completing these steps, be sure to follow the guidelines listed in the "Lock-and-Key Configuration Guidelines" section of this chapter.

SUMMARY STEPS

1. Router(config)# **access-list** *access-list-number* [**dynamic** *dynamic-name* [**timeout** *minutes*]] {**deny** | **permit**} **telnet** *source source-wildcard destination destination-wildcard*[**precedence** *precedence*] [**tos** *tos*] [**established**] [**log**]
2. Router(config)# **access-list dynamic-extend**
3. Router(config)# **interface** *type number*
4. Router(config-if)# **ip access-group** *access-list-number*
5. Router(config-if)# **exit**
6. Router(config)# **line vty** *line-number* [*ending-line-number*]
7. Do one of the following:
    - Router(config-line)# **login tacacs**
    -
    - Router(config-line)# **password** *password*
8. Do one of the following:
    - Router(config-line)# **autocommand access-enable** [**host]** [**timeout** *minutes*]
    -
    - Router# **access-enable** [**host]** [**timeout** *minutes*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **access-list** *access-list-number* [**dynamic** *dynamic-name* [**timeout** *minutes*]] {**deny** | **permit**} **telnet** *source source-wildcard*	Configures a dynamic access list, which serves as a template and placeholder for temporary access list entries.

	Command or Action	Purpose
	destination destination-wildcard[**precedence** *precedence*] [**tos** *tos*] [**established**] [**log**]	
Step 2	Router(config)# **access-list dynamic-extend**	(Optional) Extends the absolute timer of the dynamic ACL by six minutes when you open another Telnet session into the router to re-authenticate yourself using lock-and-key. Use this command if your job will run past the ACL's absolute timer.
Step 3	Router(config)# **interface** *type number*	Configures an interface and enters interface configuration mode.
Step 4	Router(config-if)# **ip access-group** *access-list-number*	Applies the access list to the interface.
Step 5	Router(config-if)# **exit**	Exits interface configuration mode and enters global configuration mode.
Step 6	Router(config)# **line vty** *line-number* [*ending-line-number*]	Defines one or more virtual terminal (VTY) ports and enters line configuration mode. If you specify multiple VTY ports, they must all be configured identically because the software hunts for available VTY ports on a round-robin basis. If you do not want to configure all your VTY ports for lock-and-key access, you can specify a group of VTY ports for lock-and-key support only.
Step 7	Do one of the following: • Router(config-line)# **login tacacs** • • Router(config-line)# **password** *password* **Example:** `Router(config-line)#` **login local** **Example:** `Router(config-line)#` **exit** **Example:** `then` **Example:** `Router(config)#` **username** *name* **password** *secret*	Configures user authentication in line or global configuration mode.

	Command or Action	Purpose
Step 8	Do one of the following: • Router(config-line)# **autocommand access-enable** [**host**] [**timeout** *minutes*] • • Router# **access-enable** [**host**] [**timeout** *minutes*]	Enables the creation of temporary access list entries in line configuration or privilege EXEC mode. Using the **autocommand** with the **access-enable** command in line configuration mode configures the system to automatically create a temporary access list entry in the dynamic access list when the host connects to the line (or lines). If the optional **host** keyword is not specified, all hosts on the entire network are allowed to set up a temporary access list entry. The dynamic access list contains the network mask to enable the new network connection. If the optional **timeout** keyword is specified, it defines the idle timeout for the temporary access list. Valid values, in minutes, range from 1 to 9999.

Verifying Lock-and-Key Configuration

You can verify that lock-and-key is successfully configured on the router by asking a user to test the connection. The user should be at a host that is permitted in the dynamic access list, and the user should have AAA authentication and authorization configured.

To test the connection, the user should Telnet to the router, allow the Telnet session to close, and then attempt to access a host on the other side of the router. This host must be one that is permitted by the dynamic access list. The user should access the host with an application that uses the IP protocol.

The following sample display illustrates what end-users might see if they are successfully authenticated. Notice that the Telnet connection is closed immediately after the password is entered and authenticated. The temporary access list entry is then created, and the host that initiated the Telnet session now has access inside the firewall.

```
Router% telnet corporate
Trying 172.21.52.1 ...
Connected to corporate.example.com.
Escape character is '^]'.
User Access Verification
Password:Connection closed by foreign host.
```

You can then use the **show access-lists** command at the router to view the dynamic access lists, which should include an additional entry permitting the user access through the router.s

Displaying Dynamic Access List Entries

You can display temporary access list entries when they are in use. After a temporary access list entry is cleared by you or by the absolute or idle timeout parameter, it can no longer be displayed. The number of matches displayed indicates the number of times the access list entry was hit.

To view dynamic access lists and any temporary access list entries that are currently established, use the following command in privileged EXEC mode:

Command	Purpose
`Router#` **show access-lists** [*access-list-number*]	Displays dynamic access lists and temporary access list entries.

Manually Deleting Dynamic Access List Entries

To manually delete a temporary access list entry, use the following command in privileged EXEC mode:

Command	Purpose
`Router#` **clear access-template** [*access-list-number* | *name*] [*dynamic-name*] [*source*] [*destination*]	Deletes a dynamic access list.

Configuration Examples for Lock-and-Key

Example Lock-and-Key with Local Authentication

This example shows how to configure lock-and-key access, with authentication occurring locally at the router. Lock-and-key is configured on the Ethernet 0 interface.

```
interface ethernet0
 ip address 172.18.23.9 255.255.255.0
 ip access-group 101 in
access-list 101 permit tcp any host 172.18.21.2 eq telnet
access-list 101 dynamic mytestlist timeout 120 permit ip any any
line vty 0
login local
autocommand access-enable timeout 5
```

The first access-list entry allows only Telnet into the router. The second access-list entry is always ignored until lock-and-key is triggered.

In the **access-list** command, the timeout is the absolute timeout. In this example, the lifetime of the mytestlist ACL is 120 minutes; that is, when a user logs in and enable the **access-enable** command, a dynamic ACL is created for 120 minutes (the maximum absolute time). The session is closed after 120 minutes, whether or not anyone is using it.

In the **access-enable**command, the timeout is the idle timeout. In this example, each time the user logs in or authenticates there is a 5-minute session. If there is no activity, the session closes in 5 minutes and the user has to reauthenticate. If the user uses the connection, the absolute time takes affect and the session closes in 120 minutes.

After a user opens a Telnet session into the router, the router will attempt to authenticate the user. If authentication is successful, the **autocommand** executes and the Telnet session terminates. The **autocommand** creates a temporary inbound access list entry at the Ethernet 0 interface, based on the second access-list entry (mytestlist). If there is no activity, this temporary entry will expire after 5 minutes, as specified by the timeout.

Example Lock-and-Key with TACACS+ Authentication

Cisco recommends that you use a TACACS+ server for authentication, as shown in the example.

The following example shows how to configure lock-and-key access, with authentication on a TACACS+ server. Lock-and-key access is configured on the BRI0 interface. Four VTY ports are defined with the password “password1”.

```
aaa authentication login default group tacacs+ enable
aaa accounting exec stop-only group tacacs+
aaa accounting network stop-only group tacacs+
enable password ciscotac
!
isdn switch-type basic-dms100
!
interface ethernet0
ip address 172.18.23.9 255.255.255.0
!
interface BRI0
 ip address 172.18.21.1 255.255.255.0
 encapsulation ppp
 dialer idle-timeout 3600
 dialer wait-for-carrier-time 100
 dialer map ip 172.18.21.2 name dialermapname
 dialer-group 1
 isdn spid1 2036333715291
 isdn spid2 2036339371566
 ppp authentication chap
 ip access-group 102 in
!
access-list 102 permit tcp any host 172.18.21.2 eq telnet
access-list 102 dynamic testlist timeout 5 permit ip any any
!
!
ip route 172.18.250.0 255.255.255.0 172.18.21.2
priority-list 1 interface BRI0 high
tacacs-server host 172.18.23.21
tacacs-server host 172.18.23.14
tacacs-server key test1
tftp-server rom alias all
!
dialer-list 1 protocol ip permit
!
line con 0
 password password1
line aux 0
 line VTY 0 4
 autocommand access-enable timeout 5
 password password1
!
```

CHAPTER 17

Configuring IP Session Filtering (Reflexive Access Lists)

This chapter describes how to configure reflexive access lists on your router. Reflexive access lists provide the ability to filter network traffic at a router, based on IP upper-layer protocol "session" information.

Restrictions on Using Reflexive Access Lists

Reflexive access lists do not work with some applications that use port numbers that change during a session. For example, if the port numbers for a return packet are different from the originating packet, the return packet will be denied, even if the packet is actually part of the same session.

The TCP application of FTP is an example of an application with changing port numbers. With reflexive access lists, if you start an FTP request from within your network, the request will not complete. Instead, you must use Passive FTP when originating requests from within your network.

Information About Reflexive Access Lists

Reflexive access lists allow IP packets to be filtered based on upper-layer session information. You can use reflexive access lists to permit IP traffic for sessions originating from within your network but to deny IP traffic for sessions originating from outside your network. This is accomplished by reflexive filtering, a kind of session filtering.

Reflexive access lists can be defined with extended named IP access lists only. You cannot define reflexive access lists with numbered or standard named IP access lists or with other protocol access lists.

You can use reflexive access lists in conjunction with other standard access lists and static extended access lists.

Benefits of Reflexive Access Lists

Reflexive access lists are an important part of securing your network against network hackers, and can be included in a firewall defense. Reflexive access lists provide a level of security against spoofing and certain denial-of-service attacks. Reflexive access lists are simple to use, and, compared to basic access lists, provide greater control over which packets enter your network.

What Is a Reflexive Access List

Reflexive access lists are similar in many ways to other access lists. Reflexive access lists contain condition statements (entries) that define criteria for permitting IP packets. These entries are evaluated in order, and when a match occurs, no more entries are evaluated.

However, reflexive access lists have significant differences from other types of access lists. Reflexive access lists contain only temporary entries; these entries are automatically created when a new IP session begins (for example, with an outbound packet), and the entries are removed when the session ends. Reflexive access lists are not themselves applied directly to an interface, but are "nested" within an extended named IP access list that is applied to the interface. (For more information about this, see the section "How to Configure Reflexive Access Lists" later in this chapter.) Also, reflexive access lists do not have the usual implicit "deny all traffic" statement at the end of the list, because of the nesting.

How Reflexive Access Lists Implement Session Filtering

With Basic Access Lists

With basic standard and static extended access lists, you can approximate session filtering by using the **established** keyword with the **permit** command. The **established** keyword filters TCP packets based on whether the ACK or RST bits are set. (Set ACK or RST bits indicate that the packet is not the first in the session, and therefore, that the packet belongs to an established session.) This filter criterion would be part of an access list applied permanently to an interface.

With Reflexive Access Lists

Reflexive access lists, however, provide a truer form of session filtering, which is much harder to spoof because more filter criteria must be matched before a packet is permitted through. (For example, source and destination addresses and port numbers are checked, not just ACK and RST bits.) Also, session filtering uses temporary filters which are removed when a session is over. This limits the hacker's attack opportunity to a smaller time window.

Moreover, the previous method of using the **established** keyword was available only for the TCP upper-layer protocol. So, for the other upper-layer protocols (such as UDP, ICMP, and so forth), you would have to either permit all incoming traffic or define all possible permissible source/destination host/port address pairs for each protocol. (Besides being an unmanageable task, this could exhaust NVRAM space.)

Where to Configure Reflexive Access Lists

Configure reflexive access lists on border routers--routers that pass traffic between an internal and external network. Often, these are firewall routers.

Note In this chapter, the words "within your network" and "internal network" refer to a network that is controlled (secured), such as your organization's intranet, or to a part of your organization's internal network that has higher security requirements than another part. "Outside your network" and "external network" refer to a network that is uncontrolled (unsecured) such as the Internet or to a part of your organization's network that is not as highly secured.

How Reflexive Access Lists Work

A reflexive access list is triggered when a new IP upper-layer session (such as TCP or UDP) is initiated from inside your network, with a packet traveling to the external network. When triggered, the reflexive access list generates a new, temporary entry. This entry will permit traffic to enter your network if the traffic is part of the session, but will not permit traffic to enter your network if the traffic is not part of the session.

For example, if an outbound TCP packet is forwarded to outside of your network, and this packet is the first packet of a TCP session, then a new, temporary reflexive access list entry will be created. This entry is added to the reflexive access list, which applies to inbound traffic. The temporary entry has characteristics as described next.

Temporary Access List Entry Characteristics

- The entry is always a **permit** entry.
- The entry specifies the same protocol (TCP) as the original outbound TCP packet.
- The entry specifies the same source and destination addresses as the original outbound TCP packet, except the addresses are swapped.
- The entry specifies the same source and destination port numbers as the original outbound TCP packet, except the port numbers are swapped.

(This entry characteristic applies only for TCP and UDP packets. Other protocols, such as ICMP and IGMP, do not have port numbers, and other criteria are specified. For example, for ICMP, type numbers are used instead.)

- Inbound TCP traffic will be evaluated against the entry, until the entry expires. If an inbound TCP packet matches the entry, the inbound packet will be forwarded into your network.
- The entry will expire (be removed) after the last packet of the session passes through the interface.
- If no packets belonging to the session are detected for a configurable length of time (the timeout period), the entry will expire.

When the Session Ends

Temporary reflexive access list entries are removed at the end of the session. For TCP sessions, the entry is removed 5 seconds after two set FIN bits are detected, or immediately after matching a TCP packet with the RST bit set. (Two set FIN bits in a session indicate that the session is about to end; the 5-second window allows the session to close gracefully. A set RST bit indicates an abrupt session close.) Or, the temporary entry is removed after no packets of the session have been detected for a configurable length of time (the timeout period).

For UDP and other protocols, the end of the session is determined differently than for TCP. Because other protocols are considered to be connectionless (sessionless) services, there is no session tracking information embedded in packets. Therefore, the end of a session is considered to be when no packets of the session have been detected for a configurable length of time (the timeout period).

Choosing an Interface Internal or External

Before you configure reflexive access lists, you must decide whether to configure reflexive access lists on an internal or external interface. You should also be sure that you have a basic understanding of the IP protocol and of access lists; specifically, you should know how to configure extended named IP access lists. To learn about configuring IP extended access lists, refer to the “Configuring IP Services” chapter of the *Cisco IOS IP Configuration Guide* .

Reflexive access lists are most commonly used with one of two basic network topologies. Determining which of these topologies is most like your own can help you decide whether to use reflexive access lists with an internal interface or with an external interface (the interface connecting to an internal network, or the interface connecting to an external network).

The first topology is shown in the figure below. In this simple topology, reflexive access lists are configured for the external interface Serial 1. This prevents IP traffic from entering the router and the internal network, unless the traffic is part of a session already established from within the internal network.

The second topology is shown in the figure below. In this topology, reflexive access lists are configured for the internal interface Ethernet 0. This allows external traffic to access the services in the Demilitarized Zone (DMZ), such as DNS services, but prevents IP traffic from entering your internal network--unless the traffic is part of a session already established from within the internal network.

Use these two example topologies to help you decide whether to configure reflexive access lists for an internal or external interface.

External Interface Configuration Task List

To configure reflexive access lists for an external interface, perform the following tasks:

1. Defining the reflexive access list(s) in an outbound IP extended named access list
2. Nesting the reflexive access list(s) in an inbound IP extended named access list
3. Setting a global timeout value

These tasks are described in the sections following the "Defining the Reflexive Access List(s)” section.

Note The defined (outbound) reflexive access list evaluates traffic traveling out of your network: if the defined reflexive access list is matched, temporary entries are created in the nested (inbound) reflexive access list. These temporary entries will then be applied to traffic traveling into your network.

Internal Interface Configuration Task List

To configure reflexive access lists for an internal interface, perform the following tasks:

1. Defining the reflexive access list(s) in an inbound IP extended named access list
2. Nesting the reflexive access list(s) in an outbound IP extended named access list
3. Setting a global timeout value

These tasks are described in the next sections.

Note The defined (inbound) reflexive access list is used to evaluate traffic traveling out of your network: if the defined reflexive access list is matched, temporary entries are created in the nested (outbound) reflexive access list. These temporary entries will then be applied to traffic traveling into your network.

Mixing Reflexive Access List Statements with Other Permit and Deny Entries

The extended IP access list that contains the reflexive access list **permit** statement can also contain other normal **permit** and **deny** statements (entries). However, as with all access lists, the order of entries is important, as explained in the next few paragraphs.

If you configure reflexive access lists for an external interface, when an outbound IP packet reaches the interface, the packet will be evaluated sequentially by each entry in the outbound access list until a match occurs.

If the packet matches an entry prior to the reflexive **permit** entry, the packet will not be evaluated by the reflexive **permit** entry, and no temporary entry will be created for the reflexive access list (reflexive filtering will not be triggered).

The outbound packet will be evaluated by the reflexive **permit** entry only if no other match occurs first. Then, if the packet matches the protocol specified in the reflexive **permit** entry, the packet is forwarded out of the interface and a corresponding temporary entry is created in the inbound reflexive access list (unless the corresponding entry already exists, indicating the outbound packet belongs to a session in progress). The temporary entry specifies criteria that permits inbound traffic only for the same session.

How to Configure Reflexive Access Lists

Defining the Reflexive Access List(s)

To define a reflexive access list, you use an entry in an extended named IP access list. This entry must use the **reflect** keyword.

- If you are configuring reflexive access lists for an external interface, the extended named IP access list should be one that is applied to outbound traffic.
- If you are configuring reflexive access lists for an internal interface, the extended named IP access list should be one that is applied to inbound traffic.
- If the extended named IP access list you just specified has never been applied to the interface, you must also apply the extended named IP access list to the interface.

SUMMARY STEPS

1. Router(config)# **ip access-list extended** *name*
2. Router(config-ext-nacl)# **permit** *protocol* **any any reflect** *name* [**timeout** *seconds*]
3. Router(config-ext-nacl)# **exit**
4. Router(config)# **interface** *type number*
5. Do one of the following:
    - Router(config-if)# **ip access-group** *name* **out**
    - Router(config-if)# **ip access-group** *name* **in**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **ip access-list extended** *name*	External interface: Specifies the outbound access list. or Internal interface: Specifies the inbound access list. (This command enters access-list configuration mode.)
Step 2	Router(config-ext-nacl)# **permit** *protocol* **any any reflect** *name* [**timeout** *seconds*]	Defines the reflexive access list using the reflexive **permit** entry. • Repeat this step for each IP upper-layer protocol; for example, you can define reflexive filtering for TCP sessions and also for UDP sessions. You can use the same *name* for multiple protocols. **Note** The reflexive list is not limited to one per ACL. It is related to each item in the ACL. You can have several reflexive lists that can be tied in to any number of items in the ACL, that are common to one input interface(or many) and evaluated on different output interface.

	Command or Action	Purpose
		For additional guidelines for this task, see the following section, "Nesting the Reflexive Access List(s)."
Step 3	Router(config-ext-nacl)# **exit**	Exits access-list configuration mode and enters global configuration mode.
Step 4	Router(config)# **interface** *type number*	Configures an interface and enters interface configuration mode.
Step 5	Do one of the following: • Router(config-if)# **ip access-group** *name* **out** • Router(config-if)# **ip access-group** *name* **in**	External interface: Applies the extended access list to the interface's outbound traffic. Internal interface: Applies the extended access list to the interface's inbound traffic.

Nesting the Reflexive Access List(s)

After you define a reflexive access list in one IP extended access list, you must "nest" the reflexive access list within a different extended named IP access list.

- If you are configuring reflexive access lists for an external interface, nest the reflexive access list within an extended named IP access list applied to inbound traffic.
- If you are configuring reflexive access lists for an internal interface, nest the reflexive access list within an extended named IP access list applied to outbound traffic.

After you nest a reflexive access list, packets heading into your internal network can be evaluated against any reflexive access list temporary entries, along with the other entries in the extended named IP access list.

Again, the order of entries is important. Normally, when a packet is evaluated against entries in an access list, the entries are evaluated in sequential order, and when a match occurs, no more entries are evaluated. With a reflexive access list nested in an extended access list, the extended access list entries are evaluated sequentially up to the nested entry, then the reflexive access list entries are evaluated sequentially, and then the remaining entries in the extended access list are evaluated sequentially. As usual, after a packet matches any of these entries, no more entries will be evaluated.

If the extended named IP access list you just specified has never been applied to the interface, you must also apply the extended named IP access list to the interface.

SUMMARY STEPS

1. Router(config)# **ip access-list extended** *name*
2. Router(config-ext-nacl)# **evaluate** *name*
3. Router(config-ext-nacl)# **exit**
4. Router(config)# **interface** *type number*
5. Do one of the following:
    - Router(config-if)# **ip access-group** *name* **in**
    - Router(config-if)# **ip access-group** *name* **out**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **ip access-list extended** *name*	External interface: Specifies the inbound access list. or Internal interface: Specifies the outbound access list. (This command enters access-list configuration mode.)
Step 2	Router(config-ext-nacl)# **evaluate** *name*	Adds an entry that "points" to the reflexive access list. Adds an entry for each reflexive access list *name* previously defined.
Step 3	Router(config-ext-nacl)# **exit**	Exits access-list configuration mode and enters global configuration mode.
Step 4	Router(config)# **interface** *type number*	Configures an interface and enters interface configuration mode.
Step 5	Do one of the following: • Router(config-if)# **ip access-group** *name* **in** • Router(config-if)# **ip access-group** *name* **out**	External interface: Applies the extended access list to the interface's inbound traffic. Internal interface: Applies the extended access list to the interface's outbound traffic.

Setting a Global Timeout Value

Reflexive access list entries expire after no packets in the session have been detected for a certain length of time (the "timeout" period). You can specify the timeout for a particular reflexive access list when you define the reflexive access list. But if you do not specify the timeout for a given reflexive access list, the list will use the global timeout value instead.

The global timeout value is 300 seconds by default. But, you can change the global timeout to a different value at any time.

To change the global timeout value, use the following command in global configuration mode:

Command	Purpose
`Router(config)#` **ip reflexive-list timeout** *seconds*	Changes the global timeout value for temporary reflexive access list entries. Use a positive integer from 0 to 2,147,483.

Configuration Examples for Reflexive Access List

Example External Interface Configuration

This example shows reflexive access lists configured for an external interface.

This configuration example permits both inbound and outbound TCP traffic at interface Serial 1, but only if the first packet (in a given session) originated from inside your network. The interface Serial 1 connects to the Internet.

Define the interface where the session-filtering configuration is to be applied:

```
interface serial 1
 description Access to the Internet via this interface
```

Apply access lists to the interface, for inbound traffic and for outbound traffic:

```
ip access-group inboundfilters in
ip access-group outboundfilters out
```

Define the outbound access list. This is the access list that evaluates all outbound traffic on interface Serial 1.

```
ip access-list extended outboundfilters
```

Define the reflexive access list tcptraffic. This entry permits all outbound TCP traffic and creates a new access list named tcptraffic. Also, when an outbound TCP packet is the first in a new session, a corresponding temporary entry will be automatically created in the reflexive access list tcptraffic.

```
permit tcp any any reflect tcptraffic
```

Define the inbound access list. This is the access list that evaluates all inbound traffic on interface Serial 1.

```
ip access-list extended inboundfilters
```

Define the inbound access list entries. This example shows Enhanced IGRP permitted on the interface. Also, no ICMP traffic is permitted. The last entry points to the reflexive access list. If a packet does not match the first two entries, the packet will be evaluated against all the entries in the reflexive access list tcptraffic.

```
permit eigrp any any
deny icmp any any
evaluate tcptraffic
```

Define the global idle timeout value for all reflexive access lists. In this example, when the reflexive access list tcptraffic was defined, no timeout was specified, so tcptraffic uses the global timeout. Therefore, if for 120 seconds there is no TCP traffic that is part of an established session, the corresponding reflexive access list entry will be removed.

```
ip reflexive-list timeout 120
```

The example configuration looks as follows:

```
interface Serial 1
 description Access to the Internet via this interface
 ip access-group inboundfilters in
 ip access-group outboundfilters out
!
ip reflexive-list timeout 120
!
ip access-list extended outboundfilters
 permit tcp any any reflect tcptraffic
!
ip access-list extended inboundfilters
 permit eigrp any any
 deny icmp any any
 evaluate tcptraffic
```

With this configuration, before any TCP sessions have been initiated the **show access-list** EXEC command displays the following:

```
Extended IP access list inboundfilters
 permit eigrp any any
 deny icmp any any
 evaluate tcptraffic
Extended IP access list outboundfilters
 permit tcp any any reflect tcptraffic
```

Notice that the reflexive access list does not appear in this output. This is because before any TCP sessions have been initiated, no traffic has triggered the reflexive access list, and the list is empty (has no entries). When empty, reflexive access lists do not show up in **show access-list** output.

After a Telnet connection is initiated from within your network to a destination outside of your network, the **show access-list** EXEC command displays the following:

```
Extended IP access list inboundfilters
 permit eigrp any any
 deny icmp any any
 evaluate tcptraffic
Extended IP access list outboundfilters
 permit tcp any any reflect tcptraffic
Reflexive IP access list tcptraffic
 permit tcp host 172.19.99.67 eq telnet host 192.168.60.185 eq 11005 (5 matches) (time
left 115 seconds)
```

Notice that the reflexive access list tcptraffic now appears and displays the temporary entry generated when the Telnet session initiated with an outbound packet.

Example Internal Interface Configuration

This is an example configuration for reflexive access lists configured for an internal interface. This example has a topology similar to the one in the figure above (shown earlier in this chapter).

This example is similar to the previous example; the only difference between this example and the previous example is that the entries for the outbound and inbound access lists are swapped. Please refer to the previous example for more details and descriptions.

```
interface Ethernet 0
 description Access from the I-net to our Internal Network via this interface
 ip access-group inboundfilters in
 ip access-group outboundfilters out
!
ip reflexive-list timeout 120
!
ip access-list extended outboundfilters
 permit eigrp any any
 deny icmp any any
 evaluate tcptraffic
!
ip access-list extended inboundfilters
 permit tcp any any reflect tcptraffic
```

CHAPTER 18

IP Access List Entry Sequence Numbering

Users can apply sequence numbers to **permit** or **deny** statements and also reorder, add, or remove such statements from a named IP access list. This feature makes revising IP access lists much easier. Prior to this feature, users could add access list entries to the end of an access list only; therefore needing to add statements anywhere except the end required reconfiguring the access list entirely.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for IP Access List Entry Sequence Numbering

- This feature does not support dynamic, reflexive, or firewall access lists.
- This feature does not support old-style numbered access lists, which existed before named access lists. Keep in mind that you can name an access list with a number, so numbers are allowed when they are entered in the standard or extended named access list (NACL) configuration mode.

Information About IP Access List Entry Sequence Numbering

Purpose of IP Access Lists

Access lists perform packet filtering to control which packets move through the network and where. Such control can help limit network traffic and restrict the access of users and devices to the network. Access lists have many uses, and therefore many commands accept a reference to an access list in their command syntax. Access lists can be used to do the following:

- Filter incoming packets on an interface.
- Filter outgoing packets on an interface.
- Restrict the contents of routing updates.
- Limit debug output based on an address or protocol.
- Control virtual terminal line access.
- Identify or classify traffic for advanced features, such as congestion avoidance, congestion management, and priority and custom queuing.
- Trigger dial-on-demand routing (DDR) calls.

How an IP Access List Works

An access list is a sequential list consisting of at least one **permit** statement and possibly one or more **deny** statements that apply to IP addresses and possibly upper-layer IP protocols. The access list has a name by which it is referenced. Many software commands accept an access list as part of their syntax.

An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the device or leaving the device, but not traffic originating at the device.

IP Access List Process and Rules

- The software tests the source or destination address or the protocol of each packet being filtered against the conditions in the access list, one condition (**permit** or **deny** statement) at a time.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the rest of the statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet and returns an ICMP Host Unreachable message.

- If no conditions match, the software drops the packet. This is because each access list ends with an unwritten or implicit **deny** statement. That is, if the packet has not been permitted by the time it was tested against each statement, it is denied.
- The access list must contain at least one **permit** statement or else all packets are denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same **permit** or **deny** statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- If an access list is referenced by name in a command, but the access list does not exist, all packets pass.
- Only one access list per interface, per protocol, per direction is allowed.
- Inbound access lists process packets arriving at the device. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, **permit** means continue to process the packet after receiving it on an inbound interface; **deny** means discard the packet.
- Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, **permit** means send it to the output buffer; **deny** means discard the packet.

Helpful Hints for Creating IP Access Lists

- Create the access list before applying it to an interface. An interface with an empty access list applied to it permits all traffic.
- Another reason to configure an access list before applying it is because if you applied a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- In order to make the purpose of individual statements more easily understood at a glance, you can write a helpful remark before or after any statement.

Source and Destination Addresses

Source and destination address fields in an IP packet are two typical fields on which to base an access list. Specify source addresses to control the packets being sent from certain networking devices or hosts. Specify destination addresses to control the packets being sent to certain networking devices or hosts.

Wildcard Mask and Implicit Wildcard Mask

When comparing the address bits in an access list entry to a packet being submitted to the access list, address filtering uses wildcard masking to determine whether to check or ignore the corresponding IP address bits. By carefully setting wildcard masks, an administrator can select one or more IP addresses for permit or deny tests.

Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.

- A wildcard mask bit 0 means check the corresponding bit value.
- A wildcard mask bit 1 means ignore that corresponding bit value.

If you do not supply a wildcard mask with a source or destination address in an access list statement, the software assumes a default wildcard mask of 0.0.0.0.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

Transport Layer Information

You can filter packets based on transport layer information, such as whether the packet is a TCP, UDP, Internet Control Message Protocol (ICMP) or Internet Group Management Protocol (IGMP) packet.

IP Access List Entry Sequence Numbering

Benefits

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If a user wanted to insert an entry (statement) in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When a user adds a new entry, the user chooses the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Sequence Numbering Behavior

- For backward compatibility with previous releases, if entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483647. If the generated sequence number exceeds this maximum number, the following message is displayed:

```
Exceeded maximum sequence number.
```

- If you enter an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- If you enter an entry that matches an already existing entry (except for the sequence number), then no changes are made.
- If you enter a sequence number that is already present, the following error message is generated:

```
Duplicate sequence number.
```

- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Distributed support is provided so that the sequence numbers of entries in the Route Processor (RP) and line card (LC) are always synchronized.
- Sequence numbers are not nvgened. That is, the sequence numbers themselves are not saved. In the event that the system is reloaded, the configured sequence numbers revert to the default sequence starting number and increment from that number. The function is provided for backward compatibility with software releases that do not support sequence numbering.
- The IP Access List Entry Sequence Numbering feature works with named standard and extended IP access lists. Because the name of an access list can be designated as a number, numbers are acceptable.

How to Use Sequence Numbers in an IP Access List

Sequencing Access-List Entries and Revising the Access List

This task shows how to assign sequence numbers to entries in a named IP access list and how to add or delete an entry to or from an access list. It is assumed a user wants to revise an access list. The context of this task is the following:

- A user need not resequence access lists for no reason; resequencing in general is optional. The resequencing step in this task is shown as required because that is one purpose of this feature and this task demonstrates the feature.
- Step 5 happens to be a **permit** statement and Step 6 happens to be a **deny** statement, but they need not be in that order.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list resequence** *access-list-name starting-sequence-number increment*
4. **ip access-list** {**standard**| **extended**} *access-list-name*
5. Do one of the following:
   - *sequence-number* **permit** *source source-wildcard*
   - *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Do one of the following:
   - *sequence-number* **deny** *source source-wildcard*
   - *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
7. Repeat Step 5 and/or Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
8. **end**
9. **show ip access-lists** *access-list-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list resequence** *access-list-name starting-sequence-number increment* **Example:** `Device(config)# ip access-list resequence kmd1 100 15`	Resequences the specified IP access list using the starting sequence number and the increment of sequence numbers. • This example resequences an access list named kmd1. The starting sequence number is 100 and the increment is 15.
Step 4	**ip access-list** {**standard**| **extended**} *access-list-name*	Specifies the IP access list by name and enters named access list configuration mode.

	Command or Action	Purpose

	Example:
	`Device(config)# ip access-list standard kmd1`
	`Device(config)# ip access-list extended kmd1`
		• If you specify **standard**, make sure you subsequently specify **permit** and/or **deny** statements using the standard access list syntax.
		• If you specify **extended**, make sure you subsequently specify **permit** and/or **deny** statements using the extended access list syntax.

Step 5	Do one of the following: • *sequence-number* **permit** *source source-wildcard* • *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-std-nacl)# 105 permit 10.5.5.5 0.0.0 255`	Specifies a permit statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • See the permit (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP). • Use the **no** *sequence-number* command to delete an entry. • As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Device(config-ext-nacl) and you would use the extended **permit** command syntax.
Step 6	Do one of the following: • *sequence-number* **deny** *source source-wildcard* • *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Device(config-std-nacl)# 105 deny 10.6.6.7 0.0.0 255`	(Optional) Specifies a deny statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • See the deny (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP). • Use the **no** *sequence-number* command to delete an entry. • As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Device(config-ext-nacl) and you would use the extended **deny** command syntax.
Step 7	Repeat Step 5 and/or Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise the access list.

Step 8	**end**
	Example:
	`Device(config-std-nacl)# end`
		(Optional) Exits the configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose

Step 9	**show ip access-lists** *access-list-name*
	
	Example:
	
	`Device# show ip access-lists kmd1`
		(Optional) Displays the contents of the IP access list.
		
		• Review the output to see that the access list includes the new entry.
		
		`Device#` **show ip access-lists kmd1**
		
		`Standard IP access list kmd1`
		
		`100 permit 10.4.4.0, wildcard bits 0.0.0.255`
		
		`105 permit 10.5.5.0, wildcard bits 0.0.0.255`
		
		`115 permit 10.0.0.0, wildcard bits 0.0.0.255`
		
		`130 permit 10.5.5.0, wildcard bits 0.0.0.255`
		
		`145 permit 10.0.0.0, wildcard bits 0.0.0.255`

What to Do Next

If your access list is not already applied to an interface or line or otherwise referenced, apply the access list. Refer to the "Configuring IP Services" chapter of the *Cisco IOS IP Configuration Guide* for information about how to apply an IP access list.

Configuration Examples for IP Access List Entry Sequence Numbering

Example: Resequencing Entries in an Access List

The following example shows access list resequencing. The starting value is 1, and increment value is 2. The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483647.

When an entry with no sequence number is entered, by default it has a sequence number of 10 more than the last entry in the access list.

```
Device# show access-list 150
Extended IP access list 150
    10 permit ip host 10.3.3.3 host 172.16.5.34
    20 permit icmp any any
    30 permit tcp any host 10.3.3.3
    40 permit ip host 10.4.4.4 any
    50 Dynamic test permit ip any any
    60 permit ip host 172.16.2.2 host 10.3.3.12
    70 permit ip host 10.3.3.3 any log
    80 permit tcp host 10.3.3.3 host 10.1.2.2
    90 permit ip host 10.3.3.3 any
    100 permit ip any any
Device(config)# ip access-list extended 150
Device(config)# ip access-list resequence 150 1 2
Device(config)# end
Device# show access-list 150
Extended IP access list 150
    1 permit ip host 10.3.3.3 host 172.16.5.34
    3 permit icmp any any
    10 permit tcp any any eq 22 log
    7 permit ip host 10.4.4.4 any
    9 Dynamic test permit ip any any
    11 permit ip host 172.16.2.2 host 10.3.3.12
    13 permit ip host 10.3.3.3 any log
    15 permit tcp host 10.3.3.3 host 10.1.2.2
    17 permit ip host 10.3.3.3 any
    19 permit ip any any
```

Example: Adding Entries with Sequence Numbers

In the following example, a new entry is added to a specified access list:

```
Device# show ip access-list
Standard IP access list tryon
2 permit 10.4.4.2, wildcard bits 0.0.255.255
5 permit 10.0.0.44, wildcard bits 0.0.0.255
10 permit 10.0.0.1, wildcard bits 0.0.0.255
20 permit 10.0.0.2, wildcard bits 0.0.0.255
Device(config)# ip access-list standard tryon
Device(config-std-nacl)# 15 permit 10.5.5.5 0.0.0.255
Device# show ip access-list
Standard IP access list tryon
2 permit 10.4.0.0, wildcard bits 0.0.255.255
5 permit 10.0.0.0, wildcard bits 0.0.0.255
10 permit 10.0.0.0, wildcard bits 0.0.0.255
15 permit 10.5.5.0, wildcard bits 0.0.0.255
20 permit 10.0.0.0, wildcard bits 0.0.0.255
```

Example: Entry without Sequence Number

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
Device(config)# ip access-list standard 1
Device(config-std-nacl)# permit 1.1.1.1 0.0.0.255
Device(config-std-nacl)# permit 2.2.2.2 0.0.0.255
Device(config-std-nacl)# permit 3.3.3.3 0.0.0.255
Device# show access-list
Standard IP access list 1
10 permit 0.0.0.0, wildcard bits 0.0.0.255
20 permit 0.0.0.0, wildcard bits 0.0.0.255
30 permit 0.0.0.0, wildcard bits 0.0.0.255
Device(config)# ip access-list standard 1
Device(config-std-nacl)# permit 4.4.4.4 0.0.0.255
Device(config-std-nacl)# end
Device# show access-list
Standard IP access list 1
10 permit 0.0.0.0, wildcard bits 0.0.0.255
20 permit 0.0.0.0, wildcard bits 0.0.0.255
```

```
30 permit 0.0.0.0, wildcard bits 0.0.0.255
40 permit 0.4.0.0, wildcard bits 0.0.0.255
```

Additional References for IP Access List Entry Sequence Numbering

The following sections provide references related to IP access lists.

Related Documents

Related Topic	Document Title
Configuring IP access lists	"Creating an IP Access List and Applying It to an Interface"
Cisco IOS commands	Cisco IOS Master Command List, All Releases
IP access list commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for IP Access List Entry Sequence Numbering

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 18: Feature Information for IP Access List Entry Sequence Numbering

Feature Name	**Releases**	**Feature Information**
IP Access List Entry Sequence Numbering		Users can apply sequence numbers to **permit** or **deny** statements and also reorder, add, or remove such statements from a named IP access list. This feature makes revising IP access lists much easier. Prior to this feature, users could add access list entries to the end of an access list only; therefore needing to add statements anywhere except the end required reconfiguring the access list entirely. In , , support was added for the Cisco Catalyst 3850 Series Switches. The following commands were introduced or modified: **deny (IP)**, **ip access-list resequence deny (IP), permit (IP)**.

CHAPTER 19

Configuring Template ACLs

When user profiles are configured using RADIUS Attribute 242 or vendor-specific attribute (VSA) Cisco-AVPairs, similar per-user access control lists (ACLs) may be replaced by a single template ACL. That is, one ACL represents many similar ACLs. By using template ACLs, you can increase the total number of per-user ACLs while minimizing the memory and Ternary Content Addressable Memory (TCAM) resources needed to support the ACLs.

In networks where each subscriber has its own ACL, it is common for the ACL to be the same for each user except for the user's IP address. The Template ACLs feature groups ACLs with many common access control elements (ACEs) into a single ACL that saves system resources.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Template ACLs

- Cisco ASR 1000 series routers
- Cisco IOS XE Release 2.4 or a later release

Restrictions for Template ACLs

Template ACLs are activated only for per-user ACLs configured through RADIUS Attribute 242 or VSA Cisco-AVPairs (ip:inacl/outacl). No other ACL types are processed by the Template ACL feature.

Template ACL functionality is available only for IPv4 ACLs.

Template ACL functionality is not available for the following types of per-user ACLs:

- Time-based ACLs
- Dynamic ACLs
- Evaluate ACLs
- Reflexive ACLs
- ACLs configured on ISG IP sessions
- IPv6 ACLs

Disabling the Template ACL Feature

When the Template ACL feature is disabled, the system replaces all existing template ACL instances with ACLs. If the system does not have enough resources (in particular TCAM resources) to setup the required number of ACLs, the system generates an error message, and the request to disable the Template ACLs feature fails.

Information About Configuring Template ACLs

Template ACL Feature Design

When the service provider uses AAA servers to configure individual ACLs for each authorized session using with RADIUS attribute 242 or VSA Cisco-AVPairs, the number of sessions can easily exceed the maximum ACL number allowed by the system.

In networks where each subscriber has an ACL, it is common for the ACL to be the same for each user except for the user's IP address. Template ACLs alleviate this problem by grouping ACLs with many common ACEs into a single ACL that compiles faster and saves system resources.

The Template ACL feature is enabled by default, and ACLs set up using the RADIUS attribute 242 or VSA Cisco-AVPairs are considered for template status.

When the Template ACL feature is enabled, the system scans and evaluates all configured per-session ACLs and then creates all required template ACLs.

Disabling Template ACLs

When the Template ACL feature is disabled, the system replaces all existing template ACL instances with ACLs. If the system does not have enough resources (in particular TCAM resources) to setup the required number of ACLs, the system generates an error message, and the request to disable the Template ACL feature fails.

Therefore, before you disable the Template ACL feature, use the **show access-list template summary** command to view the number of template ACLs in the system and ascertain if this number exceeds the system limitations.

When the template ACL feature is disabled, no new ACLS are considered for templating.

Multiple ACLs

When the Template ACL feature is enabled, the system can identify when two per-user ACLS are similar, and the system consolidates the two per-user ACLs into one template ACL.

For example, the following example shows two ACLs for two separate users:

```
ip access-list extended Virtual-Access1.1#1 (PeerIP: 10.1.1.1)
permit igmp any host 10.1.1.1
permit icmp host 10.1.1.1 any
deny ip host 10.31.66.36 host 10.1.1.1
deny tcp host 10.1.1.1 host 10.31.66.36
permit udp any host 10.1.1.1
permit udp host 10.1.1.1 any
permit udp any host 192.168.2.1
permit udp any host 192.168.222.1
permit icmp host 10.55.15.4 host 192.168.2.1
permit udp 10.22.11.0 0.0.0.255 host 192.168.211.2
permit tcp any host 192.168.222.1
permit ip host 10.55.15.4 host 192.168.2.1
permit tcp 10.22.11.0 0.0.0.255 host 192.168.211.2
ip access-list extended Virtual-Access1.1#2 (PeerIP: 10.13.11.2)
permit igmp any host 10.13.11.2
permit icmp host 10.13.11.2 any
deny ip host 10.31.66.36 host 10.13.11.2
deny tcp host 10.13.11.2 host 10.31.66.36
permit udp any host 10.13.11.2
permit udp host 10.13.11.2 any
permit udp any host 192.168.2.1
permit udp any host 192.168.222.1
permit icmp host 10.55.15.4 host 192.168.2.1
permit udp 10.22.11.0 0.0.0.255 host 192.168.211.2
permit tcp any host 192.168.222.1
permit ip host 10.55.15.4 host 192.168.2.1
permit tcp 10.22.11.0 0.0.0.255 host 192.168.211.2
```

With the Template ACL feature is enabled, the system recognizes that these two ACLs are similar, and creates a template ACL as follows:

```
ip access-list extended Template_1
permit igmp any host <PeerIP>
permit icmp host <PeerIP> any
deny ip host 10.31.66.36 host <PeerIP>
deny tcp host <PeerIP> 10.31.66.36
permit udp any host <PeerIP>
permit udp host <PeerIP> any
permit udp any host 192.168.2.1
permit udp any host 192.168.222.1
permit icmp host 10.55.15.4 host 192.168.2.1
permit udp 10.22.11.0 0.0.0.255 host 192.168.211.2
permit tcp any host 192.168.222.1
permit ip host 10.55.15.4 host 192.168.2.1
permit tcp 10.22.11.0 0.0.0.255 host 192.168.211.2
```

In this example, the peer IP address is associated as follows:

- Virtual-Access1.1#1 10.1.1.1
- Virtual-Access1.1#2 10.13.11.2

The two ACLs are consolidated into one template ACL and are referenced as follows:

Virtual-Access1.1#1 maps to Template_1(10.1.1.1)

Virtual-Access1.1#2 maps to Template_1(10.13.11.2)

VSA Cisco-AVPairs

Template ACL processing occurs for ACLs that are configured using Cisco-AVPairs. Only AVPairs that are defined using the ACL number are considered for the templating process.

To be considered for templating, AVPairs for incoming ACLs must conform to the following format:

ip:inacl#number={standard-access-control-list | extended-access-control-list}

For example: ip:inacl#10=deny ip any 10.13.16.0 0.0.0.255

To be considered for templating, AVPairs for outgoing ACLs must conform to the following format:

ip:outacl#number={standard-access-control-list | extended-access-control-list}

For example: ip:outacl#200=permit ip any any

For more information on Cisco-AVPairs, see the Cisco Vendor-Specific AVPair Attributes section of the *Cisco IOS ISG RADIUS CoA Interface Guide*.

RADIUS Attribute 242

Template ACL processing occurs for ACLs that are configured using RADIUS attribute 242. Attribute 242 has the following format for an IP data filter:

Ascend-Data-Filter = "ip <dir> <action> [dstip <dest_ipaddr\subnet_mask>] [srcp <src_ipaddr\subnet_mask>] [<proto> [dstport <cmp> <value>] [srcport <cmp> <value>] [<est>]]"

The table below describes the elements in an attribute 242 entry for an IP data filter.

Table 19: IP Data Filter Syntax Elements

Element	Description
ip	Specifies an IP filter.
<dir>	Specifies the filter direction. Possible values are **in** (filtering packets coming into the router) or **out** (filtering packets going out of the router).
<action>	Specifies the action the router should take with a packet that matches the filter. Possible values are **forward** or **drop**.

Element	Description
dstip <dest_ipaddr\subnet_mask>	Enables destination-IP-address filtering. Applies to packets whose destination address matches the value of **<dest_ipaddr>**. If a subnet mask portion of the address is present, the router compares only the masked bits. If you set **<dest_ipaddr>** to 0.0.0.0, or if this keyword is not present, the filter matches all IP packets.
srcp<src_ipaddr\subnet_mask>	Enables source-IP-address filtering. Applies to packets whose source address matches the value of **<src_ipaddr>**. If a subnet mask portion of the address is present, the router compares only the masked bits. If you set **<src_ipaddr>** to 0.0.0.0, or if this keyword is not present, the filter matches all IP packets.
<proto>	Specifies a protocol specified as a name or a number. Applies to packets whose protocol field matches this value. Possible names and numbers are **icmp** (**1**), **tcp** (**6**), **udp** (**17**), and **ospf** (**89**). If you set this value to zero (0), the filter matches any protocol.
dstport <cmp> <value>	Enables destination-port filtering. This keyword is valid only when **<proto>** is set to **tcp** (**6**) or **udp** (**17**). If you do not specify a destination port, the filter matches any port. **<cmp>** defines how to compare the specified **<value>** to the actual destination port. This value can be <, =, >, or **!**. **<value>** can be a name or a number. Possible names and numbers are **ftp-data (20)**, **ftp (21)**, **telnet (23)**, **nameserver (42)**, **domain (53)**, **tftp (69)**, **gopher (70)**, **finger (79)**, **www (80)**, **kerberos (88)**, **hostname (101)**, **nntp (119)**, **ntp (123)**, **exec (512)**, **login (513)**, **cmd (514)**, and **talk (517)**.
srcport <cmp> <value>	Enables source-port filtering. This keyword is valid only when **<proto>** is set to **tcp**(**6**)or **udp** (**17**). If you do not specify a source port, the filter matches any port. **<cmp>** defines how to compare the specified **<value>** to the actual destination port. This value can be <, =, >, or **!**. **<value>** can be a name or a number. Possible names and numbers are **ftp-data** (**20**), **ftp** (**21**), **telnet**(**23**), **nameserver**(**42**), **domain**(**53**), **tftp**(**69**), **gopher**(**70**), **finger**(**79**), **www**(**80**), **kerberos** (**88**), **hostname** (**101**), **nntp** (**119**), **ntp**(**123**), **exec** (**512**), **login** (**513**), **cmd** (**514**), and **talk** (**517**).
<est>	When set to 1, specifies that the filter matches a packet only if a TCP session is already established. This argument is valid only when **<proto>** is set to **tcp (6)**.

"RADIUS Attribute 242 IP Data Filter Entries" shows four attribute 242 IP data filter entries.

RADIUS Attribute 242 IP Data Filter Entries

```
Ascend-Data-Filter=”ip in drop”
Ascend-Data-Filter=”ip out forward tcp”
Ascend-Data-Filter=”ip out forward tcp dstip 10.0.200.3/16 srcip 10.0.200.25/16
dstport!=telnet”
Ascend-Data-Filter=”ip out forward tcp dstip 10.0.200.3/16 srcip 10.0.200.25/16 icmp”
```

How to Configure Template ACLs

If ACLs are configured using RADIUS Attribute 242 or VSA Cisco-AVPairs, template ACLs are enabled by default.

Configuring the Maximum Size of Template ACLs

By default, template ACL status is limited to ACLs with 100 or fewer rules. However, you can set this limit to a lower number. To set the maximum number of rules that an ACL may have in order to be considered as a template ACL, perform the steps in this section:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list template** *number*
4. **exit**
5. **show access-list template summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**access-list template** *number* **Example:** `Router(config)# access-list template 50`	Enables template ACL processing. Only ACLs with the specified number of rules (or fewer rules) will be considered for template status.

	Command or Action	Purpose
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**show access-list template summary** **Example:** `Router# show access-list template summary`	(Optional) Displays summary information about template ACLs.

Troubleshooting Tips

The following commands can be used to troubleshoot the Template ACL feature:

- **show access-list template**
- show platform hardware qfp active classification class-group-manager class-group client acl all
- **show platform hardware qfp active feature acl** {**control** | **node** *acl-node-id*}
- show platform software access-list

Configuration Examples for Template ACLs

Example Maximum Size of Template ACLs

The following example shows how to set the maximum number of rules that an ACL may have in order to be considered for template status to 50. Only ACLs whose number of rules is the same as or smaller than 50 are considered for template status.

```
Router> enable

Router# configure terminal

Router(config)# access-list template 50
Router(config)# exit
```

Example Showing ACL Template Summary Information

The following example shows how to view summary information for all ACLs in the system. The output from the command includes the following information:

- Maximum number of rules per template ACL

- Number of discovered active templates
- Number of ACLs replaced by those templates
- Number of elements in the Red-Black tree

```
Router# show access-list template summary
Maximum rules per template ACL = 100
Templates active = 9
Number of ACLs those templates represent = 14769
Number of tree elements = 13
```

Red-Black Tree Elements

The number of tree elements is the number of elements in the Red-Black tree. Each template has 1 unique entry in the Red-Black tree. The system calculates a cyclic redundancy check (CRC) over each ACL masking out the peer IP address and puts the CRC into the Red-Black tree. For example:

Your system has 9 templates (representing 14769 ACLs), and 13 tree elements. If each template has only 1 unique entry in the Red-Black tree, then the additional 4 tree elements means that your system contains 4 per-user ACLs that are not templated.

Example Showing ACL Template Tree Information

The following example shows how to view Red-Black tree information for all ACLs in the system.

The output from the command includes the following information:

- Name of the ACL on the Red-Black tree
- The original CRC32 value
- Number of users of the ACL
- Calculated CRC32 value

```
Router# show access-list template tree
ACL name      OrigCRC   Count  CalcCRC
4Temp_1073741891108      59DAB725   98  59DAB725
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
IP access list commands	*Cisco IOS Security Command Reference*
Configuring IP access lists	"Creating an IP Access List and Applying It to an Interface"

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for ACL Template

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 20: Feature Information for ACL Template

Feature Name	Releases	Feature Information
Template ACLs	12.2(28)SB 12.2(31)SB2 Cisco IOS XE Release 2.4	In 12.2(28)SB, this feature was introduced on the Cisco 10000 series router. In 12.2(31)SB2, support was added for the PRE3. In Cisco IOS XE Release 2.4, this feature was implemented on the Cisco ASR 1000 series routers. The following commands were introduced or modified:**access-list template, show access-list template**

CHAPTER 20

Turbo Access Control List Scalability Enhancements

The Turbo Access Control List (ACL) Scalability Enhancements feature improves overall performance on the Cisco 7304 device using a Network Services Engine (NSE) by allowing Turbo ACLs to be processed in PXF using less memory, thereby allowing more traffic traversing the Cisco 7304 device using an NSE to be PXF-accelerated. This feature also introduces user-configuration options that allow users to define the amount of memory used for Turbo ACL purposes in the Route Processor (RP) processing path.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Turbo Access Control List Scalability Enhancements

Because the portion of this feature that more expediently removes older entries works in the PXF processing path, PXF must be enabled for this particular functionality to have any benefit. PXF processing is enabled by default.

Restrictions for Turbo Access Control List Scalability Enhancements

This feature is not available for Cisco 7304 devices using an NPE-G100.

Information About Turbo Access Control List Scalability Enhancements

How Turbo ACL on the Cisco 7304 Router Using an NSE Works

With the exception that most Turbo ACL classification is PXF-accelerated on a Cisco 7304 router using an NSE-100 or an NSE-150, Turbo ACL classification on the Cisco 7304 router using an NSE-100 or NSE-150 is similar in behavior to Turbo ACL on other platforms. For information on Turbo ACL, see Turbo Access Control Lists .

For information on PXF on Cisco 7304 routers using an NSE-100 or an NSE-150, including the Turbo ACL features that are PXF-accelerated, see PXF Information for the Cisco 7304 Router .

How Turbo ACL Scalability Enhancements on the NSEs Improves Overall PXF Performance

The memory allocated in PXF for Turbo Access Control Lists (ACLs) on the NSE-100 especially is limited to the point where even modestly-sized ACL configurations cause a large amount of PXF memory to be used for Turbo ACL processing. As a result, a large amount of network traffic that should be processed through the PXF processing path is instead processed through the RP path.

This enhancement is part of a series of enhancements to improve Turbo ACL functionality on the Cisco 7304 router using the NSE-100. Specifically, this feature keeps the entries for PXF-based Turbo ACL classification current by more actively removing older entries. The older entries, which are no longer used for current traffic flows, still consume memory and, therefore, cause traffic that would normally be PXF-accelerated to instead be punted to the RP. This portion of the feature, which does not require user configuration, improves overall traffic flow on the Cisco 7304 router using an NSE by allowing more network traffic to be PXF-accelerated.

How Turbo ACL Scalability Enhancements on the NSEs Improves Overall Route Processing Performance

These Turbo ACL scalability enhancements also introduce an enhancement that allows users, via configuration commands, to configure the amount of memory reserved for ACL processing on the RP. The ability to configure the amount of memory reserved for ACL processing in the RP path gives users the option either to improve ACL processing performance in the RP path by reserving more memory for ACL processing, or to improve all other RP path functionality by reserving less memory for ACL processing.

In Cisco IOS releases not containing this feature, the amount of memory reserved for RP ACL handling is fixed.

Understanding Memory Limits for Turbo ACL Processes on the Route Processor

An NSE-150 has 2 GB of DRAM. NSE-100 RAM is user-configurable using an SDRAM SODIMM. While most NSE-100s have 512 MB of RAM, 256-MB and 128-MB SDRAM SODIMMs for the NSE-100 exist.

On a Cisco 7304 router using an NSE-150, the default memory limit for Turbo ACL processes (such as classification, compilation, and table storage) of Layer 3 and Layer 4 data in the RP path is always 256 MB. The default memory limit for Turbo ACL processes for Layer 2 data in the RP path for a Cisco 7304 router using an NSE-150 is always 128 MB.

On a Cisco 7304 router using an NSE-100, the default amount of memory reserved for Turbo ACL processes in the RP path is dependant upon the amount of SDRAM configured on the NSE-100. If the NSE has 512 MB of SDRAM or more, the default memory limit for Turbo ACL processes for Layer 3 and Layer 4 traffic processing is 256 MB. If the processor has less than 512 MB of SDRAM, the default memory limit for Turbo ACL processes for Layer 3 and Layer 4 traffic is 128 MB.

The default amount of memory reserved for Layer 2 Turbo ACL processes for a Cisco 7304 router using an NSE-100 is always 128 MB, regardless of the amount of memory configured on the processor.

To see the default amount of memory reserved for Layer 2 or for Layer 3 and Layer 4 Turbo ACL processing on your Cisco 7304 router, enter the **show access-list compiled** command. The “Mb default limit” output, which appears in both the “Compiled ACL statistics for IPv4” and “Compiled ACL statistics for Data-Link” sections of the output, shows you the default memory reservations for either Layer 2 or Layer 3 and Layer 4 Turbo ACL processing. See "Monitoring Turbo ACL Memory Usage in the Route Processing Path" for a more detailed explanation of this procedure.

To change the default amount of memory reserved for Layer 2 or Layer 3 and Layer 4 Turbo ACL processing on your Cisco 7304 router, enter the **access-list compiled** [**ipv4** | **data-link**] **limit memory** *number*command.

To restore the default amount of memory reserved for Layer 2 or Layer 3 and Layer 4 Turbo ACL processing on your Cisco 7304 router, enter the **default access-list compiled** [**ipv4** | **data-link**] **limit memory**command.

To learn more about the SDRAM SODIMMs that determine the amount of SDRAM available for Cisco 7304 routers using an NSE-100, see NSE-100 Memory Information.

Benefits

Improved Traffic Flow

This feature improves the Turbo ACL processing process in PXF by more expediently removing older entries. As a result, more Turbo ACL processing can be done in the PXF processing path, thereby allowing more router traffic to be accelerated using the PXF processing path.

Configuration of Route Processor Memory Limits for ACL Processing

This feature allows users to set the amount of memory reserved for ACL processes (such as compilation, storage, and classification) in the RP path. Users who need more memory for ACL processes now have the ability to set aside additional memory resources in the RP path for ACL processes. Users who need more more memory for other processes in the RP path now can set aside less memory for ACL processes.

How to Configure Turbo Access Control List Scalability Enhancements

It is important to note that the portion of this feature that more expediently removes older ACL entries for ACLs being processed in the PXF processing path occurs automatically without user configuration.

The following sections contain procedures for configuring memory reservations for Turbo ACL processing on the RP:

Monitoring Turbo ACL Memory Usage in the Route Processing Path

Before setting the actual memory limits for RP-based Turbo ACL usage, it may be helpful to gather information regarding the amount of memory being used for Turbo ACL usage.

To monitor your Turbo ACL memory usage in the RP path, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. show access-list compiled

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	show access-list compiled	Displays the status and condition of the Turbo ACL tables associated with each access list.

	Command or Action	Purpose
	Example: `Router# show access-list compiled`	When using this command to verify memory limitation settings for Turbo ACL processing, look for the following: • The output for **show access-list compiled** is separated for Layer 2 and for Layer 3 and Layer 4 data. Layer 3 and Layer 4 ACL compilation tables and information can be seen in the “Compiled ACL statistics for IPv4” section of the output, while Layer 2 ACL compilation tables and information can be seen in the “Compiled ACL statistics for Data-Link” section. • The “mem limits” output that shows the number of times a compile has occurred and the ACL has reached its configured limit. • The “Mb limit” output that shows the current memory limit setting. • The “Mb max memory” output that shows the maximum amount of memory the current ACL configuration could actually consume under maximum usage conditions. For additional information and an example, see "Monitoring Memory Limitations for Layer 2 or Layer 3 and Layer 4 ACL Processing".

Configuring a User-Defined Memory Limitations for Turbo ACL Processing Path

To enable memory limitations for Turbo ACL processing of Layer 3 and Layer 4 data in the RP path, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list compiled ipv4 limit memory** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**access-list compiled ipv4 limit memory** *number* **Example:** `Router(config)# access-list compiled ipv4 limit memory 300`	Specifies the limit, in megabytes, reserved for Turbo ACL instance 0, which is used for processing Layer 3 and Layer 4 data.

Removing Memory Limits for Turbo ACL Processing of Layer 3 and Layer 4 Data in the Route Processing Path

Removing all memory limits for Turbo ACL processes in the Route Processor allows all route processing memory to be used for Turbo ACL processing of Layer 3 and Layer 4 data, if necessary. It is important to note that this functionality is not used to remove a previously configured limit, even though it is a **no** form of a command.

To remove all memory limits for Turbo ACL processing for Layer 3 and Layer 4 data and to allow as much memory as needed for Layer 3 and Layer 4 Turbo ACL processing in the RP path, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no access-list compiled ipv4 limit memory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**no access-list compiled ipv4 limit memory** **Example:** `Router(config)# no access-list compiled ipv4 limit memory`	Removes any memory limits for Layer 3 and Layer 4 Turbo ACL processing, thereby allowing all available memory to be used for Layer 3 and Layer 4 Turbo ACL processing, if necessary.

Restoring the Default Memory Limits for Turbo ACL Processing of Layer 3 and 4 Data in the Route Processing Path

The default memory limit for Turbo ACL processing of Layer 3 and Layer 4 data in the RP path is always 256 MB on the NSE-150.

On the NSE-100, the default memory limit for Turbo ACL processing of Layer 3 and Layer 4 data in the RP path is dependant on the amount of memory on your NSE-100. If you have more than 512 MB of memory configured on your processor, your default memory limit for RP-based Turbo ACL processing is 256 MB. If you have less than 512 MB of memory, your default memory limit for RP-based Turbo ACL processing is 128 MB.

To restore the default RP memory limit settings for Turbo ACL processing of Layer 3 and Layer 4 traffic, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **default access-list compiled ipv4 limit memory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**default access-list compiled ipv4 limit memory** **Example:** `Router(config)# default access-list compiled ipv4 limit memory`	Restores the default memory limit setting for Layer 3 and Layer 4 Turbo ACL traffic processing. The default memory limit for Turbo ACL processing of Layer 3 and Layer 4 data in the RP path is always 256 MB on the NSE-150. On the NSE-100, the default memory limit for Turbo ACL processing of Layer 3 and Layer 4 data in the RP path is dependant on the amount of memory on your NSE-100. If you have more than 512 MB of memory configured on your processor, your default memory limit for RP-based Turbo ACL processing is 256 MB. If you have less than 512 MB of memory, your default memory limit for RP-based Turbo ACL processing is 128 MB.

Layer 2 Data in the Route Processing Path

To enable a memory limitation setting for Turbo ACL processing of Layer 2 data in the RP path, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list compiled data-link limit memory** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**access-list compiled data-link limit memory** *number* **Example:** `Router(config)# access-list compiled data-link limit memory 150`	Specifies the limit, in megabytes, reserved for Turbo ACL instance 1, which is used by the Turbo ACL algorithm to classify Layer 2 frames.

Removing Memory Limits for Turbo ACL Processing of Layer 2 Data in the Route Processing Path

Removing all memory limits for Turbo ACL processing of Layer 2 data in the Route Processor allows all route processing memory to be used for Turbo ACL processing of Layer 2 data, if necessary. It is important to note that this functionality is not used to remove a previously configured limit, even though it is a **no** form of a command.

To remove all RP-based memory limits for Turbo ACL processing for Layer 2 data and to allow as much memory as needed for Layer 2 Turbo ACL processing, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no access-list compiled data-link limit memory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**no access-list compiled data-link limit memory** **Example:** `Router(config)# no access-list compiled data-link limit memory`	Removes any memory limits for Layer 2 Turbo ACL processing, thereby allowing all available memory to be used for Layer 2 Turbo ACL processing, if necessary.

Restoring the Default Memory Limits for Turbo ACL Processing of Layer 2 Data in the Route Processing Path

The default memory limit for Turbo ACL processing of Layer 2 data in the RP processing path is 128 MB for the NSE-100 and NSE-150.

To restore the default RP-based memory limit setting for Turbo ACL processing of Layer 2 data, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **default access-list compiled data-link limit memory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**default access-list compiled data-link limit memory** **Example:** `Router(config)# default access-list compiled data-link limit memory`	Restores the default memory limit setting for Layer 2 Turbo ACL processing. The default memory limit setting for Layer 2 Turbo ACL processing is always 128 MB.

Verifying Memory Limitation Settings for Turbo ACL Processing

To verify RP-based memory limitation settings for Turbo ACL processing, you must complete the following steps.

SUMMARY STEPS

1. **enable**
2. show access-list compiled

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	show access-list compiled **Example:** `Router# show access-list compiled`	Displays the status and condition of the Turbo ACL tables associated with each access list. When using this command to verify memory limitation settings for Turbo ACL processing, look at the “Mb limit” output for both IPv4 and Data-Link. The new MB limit setting should be listed in the “Mb limit” output for IPv4 or Data-Link, depending on which memory limit was changed. For an example of the **show access-list compiled** command with these outputs highlighted, see "Example Verifying ACL Memory Limit Configurations".

Configuration Examples for Turbo Access Control List Scalability Enhancements

Example Monitoring Memory Limitations for Layer 2 or Layer 3 and Layer 4 ACL Processing

In the following example, the **show access-list compiled** command is entered.

Note the following, which are italicized in the example output:

- The output for **show access-list compiled** is separated for Layer 2 and for Layer 3 and Layer 4 data. Layer 3 and Layer 4 ACL compilation tables and information can be seen in the "Compiled ACL statistics for IPv4" section of the output, while Layer 2 ACL compilation tables and information can be seen in the "Compiled ACL statistics for Data-Link" section.
- The "mem limits" output shows the number of times a compile has occurred and the ACL has reached its configured limit. If you have reached the configured limit numerous times, you may want to consider modifying the memory limit to allow more memory. In this example, ACL memory for Layer 3 and Layer 4 data has never reached its configured limit. The same is true for Layer 2 data in this example.
- The "Mb limit" output shows the current memory limit setting. In this example, the Layer 3 and Layer 4 memory limit was previously set to 65 MB (via the **access-list compiled ipv4 limit memory 65** command), while the Layer 2 memory limit has not been changed from its default limit of 128 MB.
- The "Mb default limit" output shows the current default memory limit setting. If the **default** form of the **access-list compiled ipv4 limit memory** command or **access-list compiled data-link limit memory** command is entered, the "Mb default limit" will become the "Mb limit." In this example, the default limits are 256 MB for Layer 3 and Layer 4 data and 128 MB for Layer 2 data.
- The "Mb max memory" output shows the maximum amount of memory the current ACL configuration could actually consume under maximum usage conditions. This number is helpful for configuring memory limits for ACL processing. If you want to free up RP memory, for instance, and you have a small number of ACLs with a low "max memory," you could configure a reservation of a small amount of memory for ACL processing using the **access-list compiled** [**ipv4** | **data-link**] **limit memory** *number* command, thereby freeing up memory for other RP processes. Conversely, if you have a high memory limit, you may want to use the **access-list compiled** [**ipv4** | **data-link**] **limit memory** *number*command to commit more memory to ACL processing, or even the **no access-list compiled** [**ipv4** | **data-link**] **limit memory** command to allow as much memory as is available for ACL processing. In this example, the max memory for the current Layer 3 and Layer 4 Turbo ACL configuration data on the router is 1 MB, and the max memory for Layer 2 Turbo ACL configuration data is 0 Mb.

```
Router# show access-lists compiledCompiled ACL statistics for IPv4:
ACL State     Entries Config Fragment Redundant
102 Operational    1       1      0        0
103 Operational    1       1      0        0
104 Operational    1       1      0        0
105 Operational    1       1      0        0
106 Operational    1       1      0        0
112 Operational    1       1      0        0
ws_def_acl Operational 1 1 0 0
7 ACLs, 7 active, 1 builds, 7 entries, 1408 ms last compile
1 history updates, 2000 history entries
0 mem limits, 65 Mb limit, 256 Mb default limit, 1 Mb max memory
0 compile failures, 0 priming failures
Overflows: L1 0, L2 0, L3 0
Table expands:[9]=0 [10]=0 [11]=0 [12]=0 [13]=0 [14]=0 [15]=0
L0: 1803Kb 2/3 8/9 3/4 2/3 2/3 2/3 2/3 2/3
L1: 5Kb 3/27 3/12 2/9 2/9
L2: 4Kb 3/150 2/81
L3: 7Kb 3/250
Ex: 8Kb
Tl: 1828Kb 41 equivs (18 dynamic)
Compiled ACL statistics for Data-Link:
ACL       State       Entries Config Fragment Redundant
int-l2-0  Operational    1       1      0        0
int-l2-1  Operational    2       2      0        0
int-l2-2  Operational    3       3      0        0
int-l2-3  Operational    4       4      0        0
int-l2-4  Operational    1       1      0        0
int-l2-5  Operational  199     199      0        0
```

```
int-l2-6  Operational  200     200      0        0
int-l2-8  Operational    3       3      0        0
int-l2-10 Operational    2       2      0        0
int-l2-15 Operational    1       1      0        0
int-l2-16 Operational    2       2      0        0
int-l2-17 Operational    3       3      0        0
int-l2-18 Operational    1       1      0        0
19 ACLs, 13 active, 22 builds, 422 entries, 832 ms last compile
0 history updates, 524288 history entries
0 mem limits, 128 Mb limit, 128 Mb default limit, 0 Mb max memory
0 compile failures, 0 priming failures
Overflows: L1 3
Table expands:[3]=3
L0: 593Kb 1013/1014 2/3
L1: 86Kb 1013/1518
Ex: 191Kb
Tl: 871Kb 2028 equivs (1013 dynamic)
```

Example Reserving a Set Amount of Memory for Layer 2 ACL Processing

The following example reserves 100 MB of memory for Layer 2 ACL processing in the RP path:

```
access-list compiled data-link limit memory 100
```

Example Allowing All Available Memory to Be Used for Layer 2 ACL Processing

The following example allows Layer 2 ACL processing to use as much memory as is needed for Layer 2 ACL processing:

```
no access-list compiled data-link limit memory
```

Example Restoring the Default Amount of Memory Reserved for Layer 2 ACL Processing

The following example restores the default amount of memory reserved for Layer 2 ACL processing in the RP path:

default access-list compiled data-link limit memory

Example Reserving a Set Amount of Memory for Layer 3 and Layer 4 ACL Processing

The following example reserves 100 MB of memory for Layer 3 and Layer 4 ACL processing in the RP path:

```
access-list compiled ipv4 limit memory 100
```

Example Allowing All Available Memory to Be Used for Layer 3 and Layer 4 ACL Processing

The following example allows Layer 3 and Layer 4 ACL processing to use as much memory as is needed for Layer 3 and Layer 4 ACL data:

```
no access-list compiled ipv4 limit memory
```

Example Restoring the Default Amount of Memory Reserved for Layer 3 and Layer 4 ACL Processing

The following example restores the default amount of memory reserved for Layer 3 and Layer 4 ACL processing in the RP path:

default access-list compiled ipv4 limit memory

Example Verifying ACL Memory Limit Configurations

In the following example, a 65-MB limit has been configured for Layer 3 and Layer 4 ACL processing, while the Layer 2 ACL memory reservations have not been changed.

See the italicized output in the following example to view the changes:

```
Router# show access-lists compiledCompiled ACL statistics for IPv4:
ACL State     Entries Config Fragment Redundant
102 Operational    1       1      0        0
103 Operational    1       1      0        0
104 Operational    1       1      0        0
105 Operational    1       1      0        0
106 Operational    1       1      0        0
112 Operational    1       1      0        0
ws_def_acl Operational 1 1 0 0
7 ACLs, 7 active, 1 builds, 7 entries, 1408 ms last compile
1 history updates, 2000 history entries
0 mem limits, 65 Mb limit, 256 Mb default limit, 1 Mb max memory
0 compile failures, 0 priming failures
Overflows: L1 0, L2 0, L3 0
Table expands:[9]=0 [10]=0 [11]=0 [12]=0 [13]=0 [14]=0 [15]=0
L0: 1803Kb 2/3 8/9 3/4 2/3 2/3 2/3 2/3 2/3
L1: 5Kb 3/27 3/12 2/9 2/9
L2: 4Kb 3/150 2/81
L3: 7Kb 3/250
Ex: 8Kb
Tl: 1828Kb 41 equivs (18 dynamic)Compiled ACL statistics for Data-Link:
ACL       State       Entries Config Fragment Redundant
int-l2-0  Operational    1       1      0        0
int-l2-1  Operational    2       2      0        0
int-l2-2  Operational    3       3      0        0
int-l2-3  Operational    4       4      0        0
int-l2-4  Operational    1       1      0        0
int-l2-5  Operational  199     199      0        0
int-l2-6  Operational  200     200      0        0
int-l2-8  Operational    3       3      0        0
int-l2-10 Operational    2       2      0        0
int-l2-15 Operational    1       1      0        0
int-l2-16 Operational    2       2      0        0
int-l2-17 Operational    3       3      0        0
```

```
int-12-18 Operational    1       1      0        0
19 ACLs, 13 active, 22 builds, 422 entries, 832 ms last compile
0 history updates, 524288 history entries
0 mem limits, 128 Mb limit, 128 Mb default limit, 0 Mb max memory
0 compile failures, 0 priming failures
Overflows: L1 3
Table expands:[3]=3
L0: 593Kb 1013/1014 2/3
L1: 86Kb 1013/1518
Ex: 191Kb
T1: 871Kb 2028 equivs (1013 dynamic)
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Access Lists	"IP Access Lists" section of *Cisco IOS IP Application Services Configuration Guide*
Network Services Engines	Cisco 7304 Network Services Engine Installation and Configuration Guide
PXF	PXF Information for the Cisco 7304 Router
Turbo Access Control Lists	Turbo Access Control Lists

Standards

Standards	Title
None	--

MIBs

MIBs	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
The Cisco Technical Support & Documentation website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, tools, and technical documentation. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Feature Information for Turbo ACL Scalability Enhancements

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 21: Feature Information for Turbo ACL Scalability Enhancements

Feature Name	Releases	Feature Information
Turbo ACL Scalability Enhancements	12.2(31)SB2	The Turbo Access Control List (ACL) Scalability Enhancements feature introduced in Cisco IOS Release 12.2(31)SB2 improves overall performance on the Cisco 7304 router using a Network Services Engine (NSE) by allowing Turbo ACLs to be processed in PXF using less memory, thereby allowing more traffic traversing the Cisco 7304 router using an NSE to be PXF-accelerated. This feature also introduces user-configuration options that allow users to define the amount of memory used for Turbo ACL purposes in the Route Processor (RP) processing path. The following commands were introduced or modified: **access-list compiled data-link limit memory**, **access-list compiled ipv4 limit memory**.

Glossary

Access Control List --A list kept by routers to control access to or from the router for a number of services.

NSE --network services engine. The Cisco 7304 router has two types of processor, the NSE and the network processing engine (NPE). Two versions of the NSE exist, the NSE-100 and the NSE-150.

RP --Route Processor. One of two processing paths on a Cisco 7304 router using an NSE, with the Parallel eXpress Forwarding path being the other path. All traffic not supported in the PXF path on a Cisco 7304 router using an NSE is forwarded using the RP path.

Turbo Access Control Lists --A Turbo Access Control list is an access list that more expediently processes traffic by compiling the ACLs into a set of lookup tables while still maintaining the match requirements.

PXF --Parallel eXpress Forwarding. One of two processing paths on a Cisco 7304 router using an NSE, with the Route Processor (RP) path being the other path. The PXF processing path is used to accelerate the performance for certain supported features.

Note See Internetworking Terms and Acronyms for terms not included in this glossary.

CHAPTER 21

IPv6 Secure Neighbor Discovery

IPv6 Secure Neighbor Discovery for Cisco software is one of several features that comprise first-hop security functionality in IPv6.

IPv6 nodes use the Neighbor Discovery (ND) protocol to discover other nodes on the link, to determine their link-layer addresses to find devices, and to maintain reachability information about the paths to active neighbors. If not secured, the Neighbor Discovery protocol is vulnerable to various attacks.

Secure neighbor discovery (SeND) is designed to counter possible threats of the Neighbor Discovery protocol. SeND defines a set of neighbor discovery options and two neighbor discovery messages. SeND also defines a new autoconfiguration mechanism to establish address ownership.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for IPv6 Secure Neighbor Discovery

The SeND feature is available on crypto images because it involves using cryptographic libraries.

Configure the following before you implement SeND on a host:

- An Rivest, Shamir, and Adelman (RSA) key pair used to generate cryptographically generated addresses (CGAs) addresses on the interface.
- A SeND modifier that is computed using the RSA key pair.
- A key on the SeND interface.
- CGAs on the SeND interface.
- A Public Key Infrastructure (PKI) trustpoint with minimum content, such as the URL of the certificate server. A trust anchor certificate must be provisioned on the host.

Complete the following tasks before you configure SeND on a host or device:

- Configure the host with one or more trust anchors.
- Configure the host with an RSA key pair or configure the host with the capability to generate an RSA key pair locally. For hosts that do not establish their own authority using a trust anchor, these keys are not certified by any certificate authority (CA).
- Configure devices with RSA keys and corresponding certificate chains, or the capability to obtain certificate chains that match the host trust anchor at some level of the chain.

Information About IPv6 Secure Neighbor Discovery

IPv6 Neighbor Discovery Trust Models and Threats

There are three IPv6 neighbor discovery trust models:

- All authenticated nodes trust each other to behave correctly at the IP layer and not to send any neighbor discovery or router discovery (RD) messages that contain false information. This model represents a situation in which the nodes are under a single administration and form a closed or semiclosed group. A corporate intranet is an example of this model.
- A device is trusted by the other nodes in the network to be a legitimate device that routes packets between the local network and any connected external networks. This device is trusted to behave correctly at the IP layer and not to send any neighbor discovery or RD messages that contain false information. This model represents a public network run by an operator. The clients pay the operator, have the operator's credentials, and trust the operator to provide the IP forwarding service. The clients do not trust each other to behave correctly; any other client node must be considered able to send falsified neighbor discovery and RD messages.
- Nodes do not trust each other at the IP layer. This model is considered suitable when a trusted network operator is not available.

Nodes on the same link use neighbor discovery to detect each other's presence and link-layer addresses, to find devices, and to maintain reachability information about paths to active neighbors. Neighbor discovery is used by both hosts and devices.

SeND Protocol

The SeND protocol counters ND threats. It defines a set of ND options, and two ND messages, Certification Path Solicitation (CPS) and Certification Path Answer (CPA). It also defines an autoconfiguration mechanism to be used in conjunction with these ND options to establish address ownership.

SeND defines the mechanisms defined in the following sections for securing ND:

Cryptographically Generated Addresses in SeND

CGAs are IPv6 addresses generated from the cryptographic hash of a public key and auxiliary parameters. CGAs securely associate a cryptographic public key with an IPv6 address in the SeND protocol.

The node generating a CGA address must first obtain an RSA key pair (SeND uses an RSA public/private key pair). The node then computes the interface identifier of the IPv6 address and appends the result to the prefix to form the CGA address.

CGA address generation is a one-time event. A valid CGA cannot be spoofed, and the message must be signed with the private key that matches the public key used for CGA generation. A user cannot replay the complete SeND message (including the CGA address, CGA parameters, and CGA signature) because the signature has only a limited lifetime.

Authorization Delegation Discovery

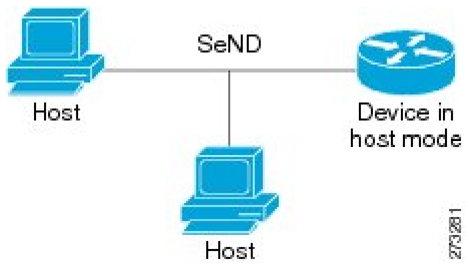
Authorization delegation discovery is used to certify the authority of devices by using a trust anchor. A trust anchor is a third party that the host trusts and to which the device has a certification path. At a basic level, the device is certified by the trust anchor. In a more complex environment, the device is certified by a user that is certified by the trust anchor. In addition to certifying the device identity (or the right for a node to act as a device), the certification path contains information about prefixes that a device is allowed to advertise in RAs. Authorization delegation discovery enables a node to adopt a device as its default device.

SeND Deployment Models

Host-to-Host Deployment Without a Trust Anchor

Deployment for SeND between hosts is straightforward. The hosts can generate a pair of RSA keys locally, autoconfigure their CGA addresses, and use them to validate their sender authority, rather than using a trust anchor to establish sender authority. The figure below illustrates this model.

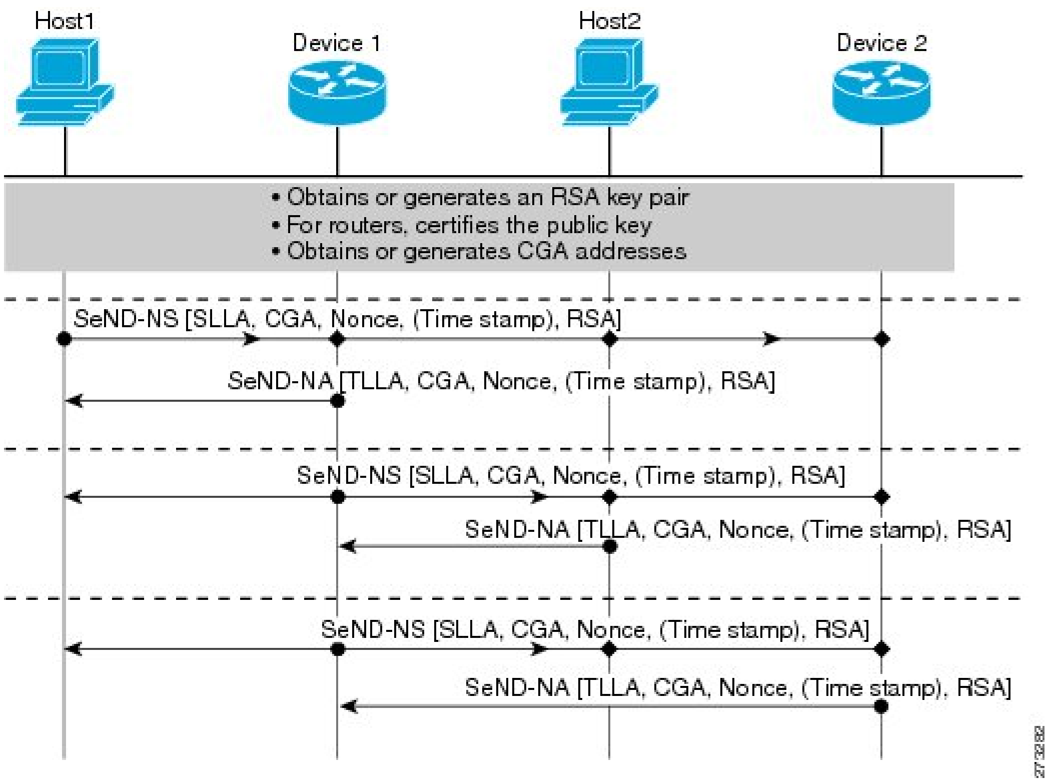
Figure 5: Host-to-Host Deployment Model



Neighbor Solicitation Flow

In a neighbor solicitation scenario, hosts and devices in host mode exchange neighbor solicitations and neighbor advertisements. These neighbor solicitations and neighbor advertisements are secured with CGA addresses and CGA options, and have nonce, time stamp, and RSA neighbor discovery options. The figure below illustrates this scenario.

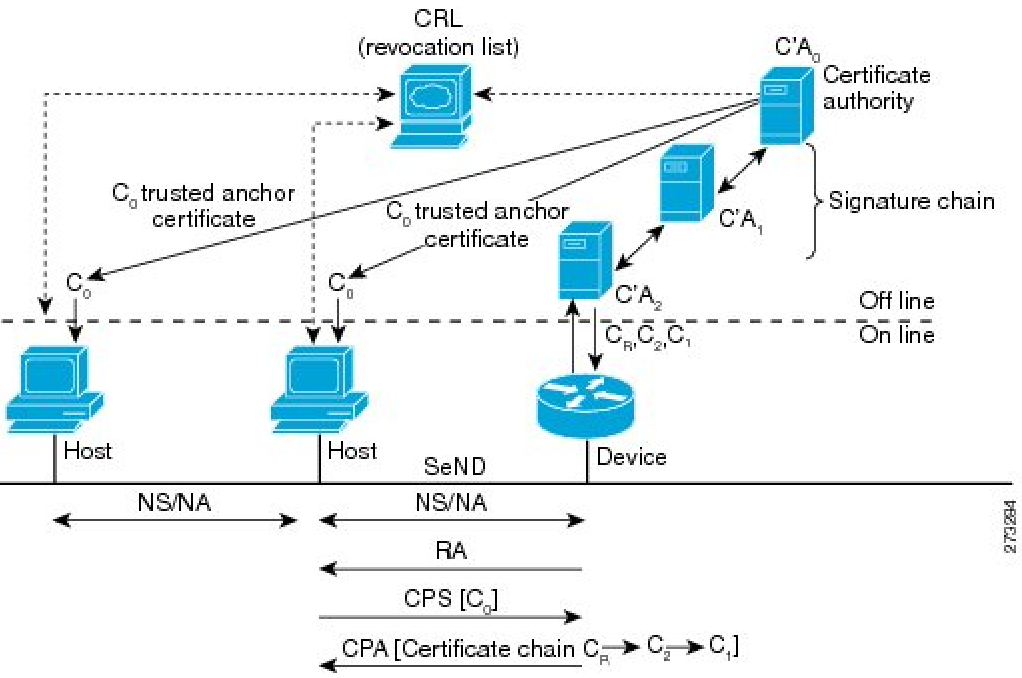
Figure 6: Neighbor Solicitation Flow


Host1
Device 1
Host2
Device 2
• Obtains or generates an RSA key pair
• For routers, certifies the public key
• Obtains or generates CGA addresses
SeND-NS [SLLA, CGA, Nonce, (Time stamp), RSA]
SeND-NA [TLLA, CGA, Nonce, (Time stamp), RSA]
SeND-NS [SLLA, CGA, Nonce, (Time stamp), RSA]
SeND-NA [TLLA, CGA, Nonce, (Time stamp), RSA]
SeND-NS [SLLA, CGA, Nonce, (Time stamp), RSA]
SeND-NA [TLLA, CGA, Nonce, (Time stamp), RSA]
273282

Host-Device Deployment Model

In many cases, hosts will not have access to the infrastructure that enables them to obtain and announce their certificates. In these situations, hosts secure their relationships using CGA, and secure their relationships with devices using trusted anchors. When using RAs, devices must be authenticated through a trust anchor. The figure below illustrates this scenario.

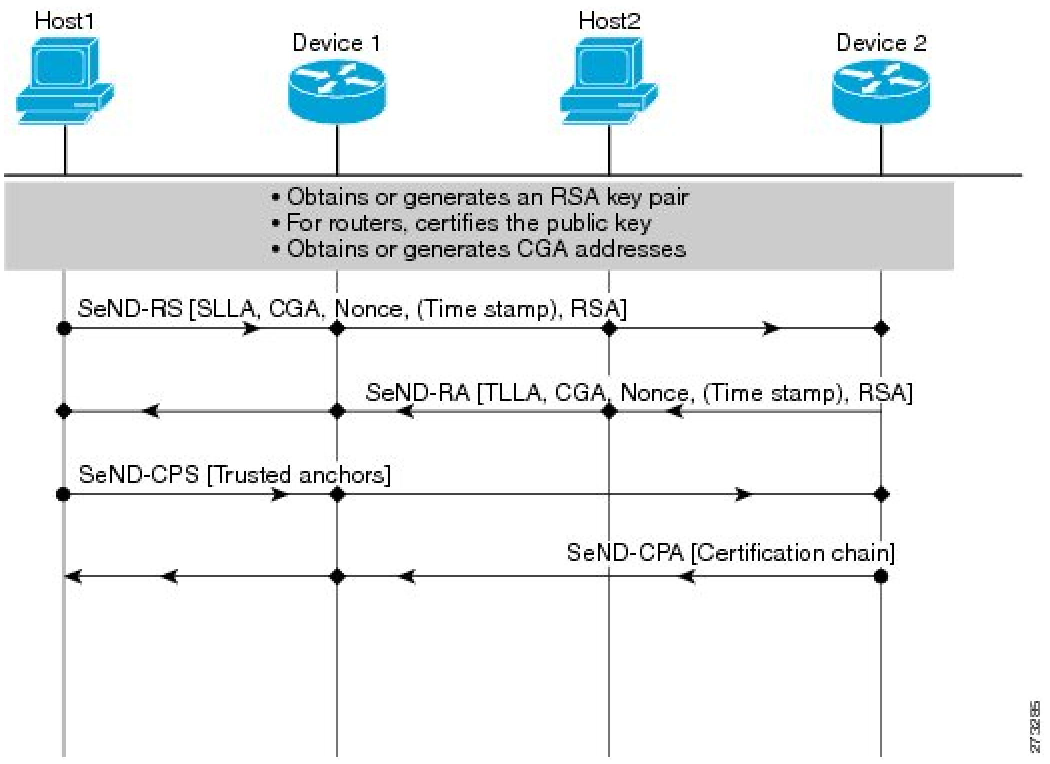
Figure 7: Host-Device Deployment Model



RAs and Certificate Path Flows

The figure below shows the certificate exchange performed using certification path solicitation CPS/CPA SeND messages. In the illustration, Router 1 is certified (using an X.509 certificate) by its own certification authority (CA). The CA itself (CA2) is certified by its own CA (certificates C2), and so on, up to a CA (CA0) that the hosts trusts. The certificate CR contains IP extensions per RFC 3779, which describes which prefix ranges the Router 1 is allowed to announce (in RAs). This prefix range, certified by CA2, is a subset of CA2's own range, certified by CA1, and so on. Part of the validation process when a certification chain is received consists of validating the certification chain and the consistency of nested prefix ranges.

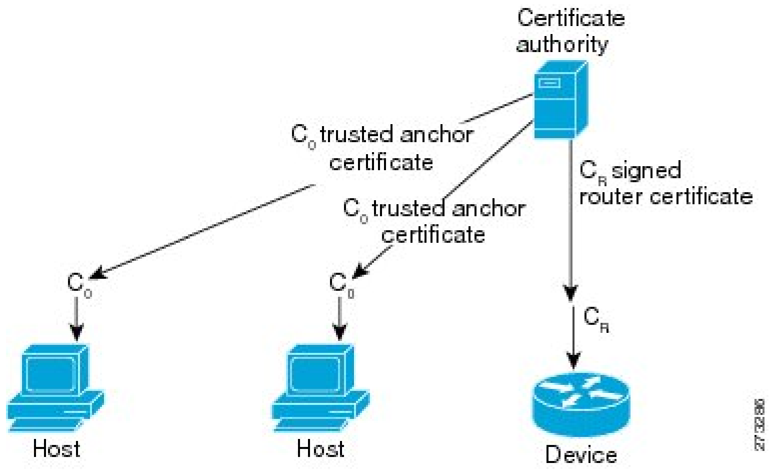
Figure 8: RAs and Certificate Path Flows


Host1
Device 1
Host2
Device 2
• Obtains or generates an RSA key pair
• For routers, certifies the public key
• Obtains or generates CGA addresses
SeND-RS [SLLA, CGA, Nonce, (Time stamp), RSA]
SeND-RA [TLLA, CGA, Nonce, (Time stamp), RSA]
SeND-CPS [Trusted anchors]
SeND-CPA [Certification chain]
273285

Single CA Deployment Model

The deployment model shown in previously can be simplified in an environment where both hosts and devices trust a single CA such as the Cisco certification server (CS). The figure below illustrates this model.

Figure 9: Single CA Deployment Model


Certificate authority
C0 trusted anchor certificate
C0 trusted anchor certificate
CR signed router certificate
C0
C0
CR
Host
Host
Device
273286

How to Configure IPv6 Secure Neighbor Discovery

Certificate servers are used to grant certificates after validating or certifying key pairs. A tool for granting certificates is mandatory in any SeND deployment. However, few certificate servers support granting certificates that contain IP extensions. Cisco certificate servers support every kind of certificate, including certificates containing IP extensions.

SeND is available in host mode. The set of available functions on a host are a subset of SeND functionality. CGA is fully available, and the prefix authorization delegation is supported on the host side (the sending CPS and receiving CPA).

SeND is also available in device mode. Use the **ipv6 unicast-routing** command to configure a node to a device. To implement SeND, configure devices with the same elements as that of the host. The devices will need to retrieve certificates of their own from a certificate server. The RSA key and subject name of the trustpoint are used to retrieve certificates from a certificate server. Once the certificate has been obtained and uploaded, the device generates a certificate request to the certificate server and installs the certificate.

Hosts and devices must either retrieve or generate their CGAs when they are booted. Typically, devices autoconfigure their CGAs once and save them (along with the key pair used in the CGA operation) in their permanent storage. At a minimum, link-local addresses on a SeND interface should be CGAs. Additionally, global addresses can be CGAs.

Configuring Certificate Servers to Enable SeND

Hosts and devices must be configured with RSA key pairs and corresponding certificate chains before the SeND parameters are configured. Perform the following task to configure the certificate server to grant certificates. Once the certificate server is configured, other parameters for the certificate server can be configured.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http server**
4. **crypto pki trustpoint** *name*
5. **ip-extension** [**multicast** | **unicast**] {**inherit** [**ipv4** | **ipv6**] | **prefix** *ipaddress* | **range** *min-ipaddress max-ipaddress*}
6. **revocation-check** {**crl** | **none** | **ocsp**}
7. **exit**
8. **crypto pki server** *name*
9. **grant auto**
10. **cdp-url** *url-name*
11. **no shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip http server** **Example:** `Device(config)# ip http server`	Configures the HTTP server.
Step 4	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint name1`	(Optional) Declares the trustpoint that your certificate server should use, and enters ca-trustpoint configuration mode. • If you plan to use X.509 IP extensions, use this command. To automatically generate a CS trustpoint, go to Step 8.

	Command or Action	Purpose

Step 5	**ip-extension** [**multicast** | **unicast**] {**inherit** [**ipv4** | **ipv6**] | **prefix** *ipaddress* | **range** *min-ipaddress max-ipaddress*}
	
	Example:
	
	`Device(ca-trustpoint)# ip-extension prefix 2001:100::/32`
		(Optional) Specifies that the IP extensions are included in a certificate request either for enrollment or generation of a CA for the Cisco CA.

Step 6	**revocation-check** {**crl** | **none** | **ocsp**}
	
	Example:
	
	`Device(ca-trustpoint)# revocation-check crl`
		(Optional) Sets a method for revocation checking.

Step 7	**exit** **Example:** `Device(ca-trustpoint)# exit`	Returns to global configuration mode.
Step 8	**crypto pki server** *name* **Example:** `Device(config)# crypto pki server server1`	Configures the PKI server, and places the device in server configuration mode.
Step 9	**grant auto** **Example:** `Device(config-server)# grant auto`	(Optional) Grants all certificate requests automatically.
Step 10	**cdp-url** *url-name* **Example:** `Device(config-server)# cdp-url http://10.165.202.129/server1.crl`	(Optional) Sets the URL name if the host is using a certificate revocation list (CRL).
Step 11	**no shutdown** **Example:** `Device(config-server)# no shutdown`	Enables the certificate server.

Configuring a Host to Enable SeND

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename*:] [**on** *devicename*:]
4. **ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
5. **crypto pki trustpoint** *name*
6. **enrollment** [**mode**] [**retry period** *minutes*] [**retry count** *number*] **url** *url* [**pem**]
7. **revocation-check** {**crl** | **none** | **ocsp**}
8. **exit**
9. **crypto pki authenticate** *name*
10. **ipv6 nd secured sec-level minimum** *value*
11. **interface** *type number*
12. **ipv6 cga rsakeypair** *key-label*
13. **ipv6 address** *ipv6-address* / *prefix-length* **link-local cga**
14. **ipv6 nd secured trustanchor** *trustanchor-name*
15. **ipv6 nd secured timestamp** {**delta** *value* | **fuzz** *value*}
16. **exit**
17. **ipv6 nd secured full-secure**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename*:] [**on** *devicename*:]	Configures the RSA key.

	Command or Action	Purpose
	Example: `Device(config)# crypto key generate rsa label SEND modulus 1024`	

Step 4	**ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
	Example:
	`Device(config)# ipv6 cga modifier rsakeypair SEND sec-level 1`
		Enables the RSA key to be used by SeND (generates the modifier).

Step 5	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint SEND`	Specifies the node trustpoint and enters ca-trustpoint configuration mode.
Step 6	**enrollment** [**mode**] [**retry period** *minutes*] [**retry count** *number*] **url** *url* [**pem**] **Example:** `Device(ca-trustpoint)# enrollment url http://10.165.200.254`	Specifies the enrollment parameters of a CA.

Step 7	**revocation-check** {**crl** | **none** | **ocsp**}
	Example:
	`Device(ca-trustpoint)# revocation-check none`
		Sets a method of revocation.

Step 8	**exit** **Example:** `Device(ca-trustpoint)# exit`	Returns to global configuration mode.
Step 9	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate SEND`	Authenticates the certification authority by getting the certificate of the CA.
Step 10	**ipv6 nd secured sec-level minimum** *value* **Example:** `Device(config)# ipv6 nd secured sec-level minimum 1`	(Optional) Configures CGA.

	Command or Action	Purpose
Step 11	**interface** *type number* **Example:** `Device(config)# interface fastethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 12	**ipv6 cga rsakeypair** *key-label* **Example:** `Device(config-if)# ipv6 cga rsakeypair SEND`	(Optional) Configures CGA on interfaces.
Step 13	**ipv6 address** *ipv6-address* / *prefix-length* **link-local cga** **Example:** `Device(config-if)# ipv6 address FE80::260:3EFF:FE11:6770/23 link-local cga`	Configures an IPv6 link-local address for the interface, and enables IPv6 processing on the interface.
Step 14	**ipv6 nd secured trustanchor** *trustanchor-name* **Example:** `Device(config-if)# ipv6 nd secured trustanchor SEND`	(Optional) Configures trusted anchors to be preferred for certificate validation.

Step 15	**ipv6 nd secured timestamp** {**delta** *value* | **fuzz** *value*}
	
	Example:
	
	`Device(config-if)# ipv6 nd secured timestamp delta 300`
		(Optional) Configures the timing parameters.

Step 16	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 17	**ipv6 nd secured full-secure** **Example:** `Device(config)# ipv6 nd secured full-secure`	(Optional) Configures general SeND parameters.

Configuring a Device to Enable SeND

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename***:**] [**on** *devicename***:**]
4. **ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
5. **crypto pki trustpoint** *name*
6. **subject-name** [**attr** *tag*] [**eq** | **ne** | **co** | **nc**] *string*
7. **rsakeypair** *key-label*
8. **revocation-check** {**crl** | **none** | **ocsp**}
9. **exit**
10. **crypto pki authenticate** *name*
11. **crypto pki enroll** *name*
12. **ipv6 nd secured sec-level minimum** *value*
13. **interface** *type number*
14. **ipv6 cga rsakeypair** *key-label*
15. **ipv6 address** *ipv6-address* **link-local cga**
16. **ipv6 nd secured trustanchor** *trustpoint-name*
17. **ipv6 nd secured timestamp** {**delta** *value* | **fuzz** *value*}
18. **exit**
19. **ipv6 nd secured full-secure**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename***:**] [**on** *devicename***:**]	Configures the RSA key.

	Command or Action	Purpose
	Example: `Device(config)# crypto key generate rsa label SEND modulus 1024`	

Step 4	**ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
	Example:
	`Device(config)# ipv6 cga modifier rsakeypair SEND sec-level 1`
		Enables the RSA key to be used by SeND (generates the modifier).

Step 5	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint SEND`	Configures PKI for a single or multiple-tier CA, specifies the device trustpoint, and places the device in ca-trustpoint configuration mode.
Step 6	**subject-name** [**attr** *tag*] [**eq** | **ne** | **co** | **nc**] *string* **Example:** `Device(ca-trustpoint)# subject-name C=FR, ST=PACA, L=Example, O=Cisco, OU=NSSTG, CN=device`	Creates a rule entry.
Step 7	**rsakeypair** *key-label* **Example:** `Device(ca-trustpoint)# rsakeypair SEND`	Binds the RSA key pair for SeND.

Step 8	**revocation-check** {**crl** | **none** | **ocsp**}
	Example:
	`Device(ca-trustpoint)# revocation-check none`
		Sets one or more methods of revocation.

Step 9	**exit** **Example:** `Device(ca-trustpoint)# exit`	Returns to global configuration mode.
Step 10	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate SEND`	Authenticates the certification authority by getting the certificate of the CA.

	Command or Action	Purpose
Step 11	**crypto pki enroll** *name* **Example:** `Device(config)# crypto pki enroll SEND`	Obtains the certificates for the device from the CA.
Step 12	**ipv6 nd secured sec-level minimum** *value* **Example:** `Device(config)# ipv6 nd secured sec-level minimum 1`	(Optional) Configures CGA and provides additional parameters such as security level and key size.
Step 13	**interface** *type number* **Example:** `Device(config)# interface fastethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 14	**ipv6 cga rsakeypair** *key-label* **Example:** `Device(config-if)# ipv6 cga rsakeypair SEND`	(Optional) Configures CGA on interfaces.
Step 15	**ipv6 address** *ipv6-address* **link-local cga** **Example:** `Device(config-if)# ipv6 address fe80::link-local cga`	Configures an IPv6 link-local address for the interface and enables IPv6 processing on the interface.
Step 16	**ipv6 nd secured trustanchor** *trustpoint-name* **Example:** `Device(config-if)# ipv6 nd secured trustanchor SEND`	(Optional) Configures trusted anchors to be preferred for certificate validation.

Step 17	**ipv6 nd secured timestamp** {**delta** *value* | **fuzz** *value*}
	
	Example:
	
	`Device(config-if)# ipv6 nd secured timestamp delta 300`
		(Optional) Configures the timing parameters.

Step 18	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 19	**ipv6 nd secured full-secure** **Example:** `Device(config)# ipv6 nd secured full-secure`	(Optional) Configures general SeND parameters, such as secure mode and authorization method.

Generating the RSA Key Pair and CGA Modifier for the Key Pair

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa** [**general-keys** \| **usage-keys** \| **signature** \| **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename*:] [**on** *devicename*:]
4. **ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** \| **1**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename*:] [**on** *devicename*:] **Example:** `Device(config)# crypto key generate rsa label SEND`	Generates RSA key pairs.

	Command or Action	Purpose

Step 4	**ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
	
	Example:
	
	`Device(config)# ipv6 cga modifier rsakeypair SEND sec-level 1`
		Generates the CGA modifier for a specified RSA key, which enables the key to be used by SeND.

Configuring Certificate Enrollment for a PKI

Certificate enrollment, which is the process of obtaining a certificate from a CA, occurs between the end host that requests the certificate and the CA. Each peer that participates in the PKI must enroll with a CA. In IPv6, you can autoenroll or manually enroll the device certificate.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *name*
4. **subject-name** *x.500-name*
5. **enrollment** [**mode**] [**retry period** *minutes*] [**retry count** *number*] **url** *url* [**pem**]
6. **serial-number** [**none**]
7. **auto-enroll** [*percent*] [**regenerate**]
8. **password** *string*
9. **rsakeypair** *key-label* [*key-size* [*encryption-key-size*]]
10. **fingerprint** *ca-fingerprint*
11. **ip-extension** [**multicast** | **unicast**] {**inherit** [**ipv4** | **ipv6**] | **prefix** *ipaddress* | **range** *min-ipaddress max-ipaddress*}
12. **exit**
13. **crypto pki authenticate** *name*
14. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint trustpoint1`	Declares the trustpoint that your device should use and enters ca-trustpoint configuration mode.
Step 4	**subject-name** *x.500-name* **Example:** `Device(ca-trustpoint)# subject-name name1`	Specifies the subject name in the certificate request.
Step 5	**enrollment** [**mode**] [**retry period** *minutes*] [**retry count** *number*] **url** *url* [**pem**] **Example:** `Device(ca-trustpoint)# enrollment url http://name1.example.com`	Specifies the URL of the CA on which your device should send certificate requests.
Step 6	**serial-number** [**none**] **Example:** `Device(ca-trustpoint)# serial-number`	(Optional) Specifies the device serial number in the certificate request.
Step 7	**auto-enroll** [*percent*] [**regenerate**] **Example:** `Device(ca-trustpoint)# auto-enroll`	(Optional) Enables autoenrollment, allowing you to automatically request a device certificate from the CA.
Step 8	**password** *string* **Example:** `Device(ca-trustpoint)# password password1`	(Optional) Specifies the revocation password for the certificate.
Step 9	**rsakeypair** *key-label* [*key-size* [*encryption-key-size*]] **Example:** `Device(ca-trustpoint)# rsakeypair SEND`	Specifies which key pair to associate with the certificate.

	Command or Action	Purpose
Step 10	**fingerprint** *ca-fingerprint* **Example:** `Device(ca-trustpoint)# fingerprint 12EF53FA 355CD23E 12EF53FA 355CD23E`	(Optional) Specifies a fingerprint that can be matched against the fingerprint of a CA certificate during authentication.

Step 11	**ip-extension** [**multicast** | **unicast**] {**inherit** [**ipv4** | **ipv6**] | **prefix** *ipaddress* | **range** *min-ipaddress max-ipaddress*}
	
	Example:
	
	`Device(ca-trustpoint)# ip-extension unicast prefix 2001:100:1://48`
		Adds IP extensions (IPv6 prefixes or range) to verifythe prefix list the device is allowed to advertise.

Step 12	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode, and returns to global configuration mode.
Step 13	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate name1`	Retrieves and authenticates the CA certificate. • This command is optional if the CA certificate is already loaded into the configuration.
Step 14	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring a Cryptographically Generated Address

Configuring General CGA Parameters

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 nd secured sec-level** [**minimum** *value*]
4. **ipv6 nd secured key-length** [[**minimum** | **maximum**] *value*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 nd secured sec-level** [**minimum** *value*] **Example:** `Device(config)# ipv6 nd secured sec-level minimum 1`	Configures the SeND security level.
Step 4	**ipv6 nd secured key-length** [[**minimum** | **maximum**] *value*] **Example:** `Device(config)# ipv6 nd secured key-length minimum 512`	Configures SeND key-length options.

Configuring CGA Address Generation on an Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 cga rsakeypair** *key-label*
5. **ipv6 address** {*ipv6-address/prefix-length* [**cga**] | *prefix-name sub-bits/prefix-length* [**cga**]}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface Ethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 4	**ipv6 cga rsakeypair** *key-label* **Example:** `Device(config-if)# ipv6 cga rsakeypair SEND`	Specifies which RSA key pair should be used on a specified interface.

Step 5	**ipv6 address** {*ipv6-address*/*prefix-length* [**cga**] | *prefix-name sub-bits*/*prefix-length* [**cga**]}
	Example:
	`Device(config-if)# ipv6 address 2001:0DB8:1:1::/64 cga`
		Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface.
		• The **cga** keyword generates a CGA address.
		Note The CGA link-local addresses must be configured by using the **ipv6 address link-local** command.

Configuring SeND Parameters

Configuring the SeND Trustpoint

The key pair used to generate the CGA addresses on an interface must be certified by the CA and the certificate sent on demand over the SeND protocol. One RSA key pair and associated certificate is enough for SeND to operate; however, users may use several keys, identified by different labels. SeND and CGA refer to a key directly by label or indirectly by trustpoint.

Multiple steps are required to bind SeND to a trustpoint. First, a key pair is generated. Then the device refers to it in a trustpoint, and next the SeND interface configuration points to the trustpoint. There are two reasons for the multiple steps:

- The same key pair can be used on several SeND interfaces.
- The trustpoint contains additional information, such as the certificate, required for SeND to perform authorization delegation.

A CA certificate must be uploaded for the referred trustpoint, which is a trusted anchor.

Several trustpoints, pointing to the same RSA keys, can be configured on a given interface. This function is useful if different hosts have different trusted anchors (that is, CAs that they trust). The device can then provide each host with the certificate signed by the CA it trusts.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename*:] [**on** *devicename*:]
4. **ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
5. **crypto pki trustpoint** *name*
6. **subject-name** [*x.500-name*]
7. **rsakeypair** *key-label* [*key-size* [*encryption-key-size*]]
8. **enrollment terminal** [**pem**]
9. **ip-extension** [**multicast** | **unicast**] {**inherit** [**ipv4** | **ipv6**] | **prefix** *ipaddress* | **range** *min-ipaddress max-ipaddress*}
10. **exit**
11. **crypto pki authenticate** *name*
12. **crypto pki enroll** *name*
13. **crypto pki import** *name* **certificate**
14. **interface** *type number*
15. **ipv6 nd secured trustpoint** *trustpoint-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename*:] [**on** *devicename*:]	Generates RSA key pairs.

	Command or Action	Purpose
	Example: `Device(config)# crypto key generate rsa label SEND`	

Step 4	**ipv6 cga modifier rsakeypair** *key-label* **sec-level** {**0** | **1**}
	Example:
	`Device(config)# ipv6 cga modifier rsakeypair SEND sec-level 1`
		Generates the CGA modifier for a specified RSA key, which enables the key to be used by SeND.

Step 5	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint trustpoint1`	Defines the trustpoint that the device should use, and enters ca-trustpoint configuration mode.
Step 6	**subject-name** [*x.500-name*] **Example:** `Device(ca-trustpoint)# subject-name name1`	Specifies the subject name in the certificate request.
Step 7	**rsakeypair** *key-label* [*key-size* [*encryption-key-size*]] **Example:** `Device(ca-trustpoint)# rsakeypair SEND`	Specifies which key pair to associate with the certificate.
Step 8	**enrollment terminal** [**pem**] **Example:** `Device(ca-trustpoint)# enrollment terminal`	Specifies manual cut-and-paste certificate enrollment.

Step 9	**ip-extension** [**multicast** | **unicast**] {**inherit** [**ipv4** | **ipv6**] | **prefix** *ipaddress* | **range** *min-ipaddress max-ipaddress*}
	Example:
	`Device(ca-trustpoint)# ip-extension unicast prefix 2001:100:1://48`
		Adds IP extensions to the device certificate request.

Step 10	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode, and returns to global configuration mode.

	Command or Action	Purpose
Step 11	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate trustpoint1`	Authenticates the certification authority by getting the certificate of the CA.
Step 12	**crypto pki enroll** *name* **Example:** `Device(config)# crypto pki enroll trustpoint1`	Obtains the certificates for your device from the CA.
Step 13	**crypto pki import** *name* **certificate** **Example:** `Device(config)# crypto pki import trustpoint1 certificate`	Imports a certificate manually using TFTP or the cut-and-paste method at the terminal.
Step 14	**interface** *type number* **Example:** `Device(config)# interface Ethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 15	**ipv6 nd secured trustpoint** *trustpoint-name* **Example:** `Device(config-if)# ipv6 nd secured trustpoint trustpoint1`	Enables SeND on an interface, and specifies which trustpoint should be used.

Configuring SeND Trust Anchors on the Interface

As soon as SeND is bound to a trustpoint on an interface, this trustpoint is also a trust anchor. A trust anchor configuration consists of the following items:

- A public key signature algorithm and associated public key, which may include parameters
- A name
- An optional public key identifier
- An optional list of address ranges for which the trust anchor is authorized

The trust anchor configuration is accomplished by binding SeND to one or several PKI trustpoints. PKI is used to upload the corresponding certificates, which contain the required parameters, such as name and key.

This optional task allows you to select trust anchors listed in the CPS when requesting for a certificate.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *name*
4. **enrollment terminal** [**pem**]
5. **exit**
6. **crypto pki authenticate** *name*
7. **interface** *type number*
8. **ipv6 nd secured trustanchor** *trustanchor-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint anchor1`	Defines the trustpoint for the device to use, and enters ca-trustpoint configuration mode.
Step 4	**enrollment terminal** [**pem**] **Example:** `Device(ca-trustpoint)# enrollment terminal`	Specifies manual cut-and-paste certificate enrollment.
Step 5	**exit** **Example:** `Device(ca-trustpoint)# exit`	Returns to global configuration.
Step 6	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate anchor1`	Authenticates the certification authority by getting the certificate of the CA.

	Command or Action	Purpose
Step 7	**interface** *type number* **Example:** `Device(config)# interface Ethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 8	**ipv6 nd secured trustanchor** *trustanchor-name* **Example:** `Device(config-if)# ipv6 nd secured trustanchor anchor1`	Configures a trusted anchor on an interface, and binds SeND to a trustpoint.

Configuring Secured and Nonsecured Neighbor Discovery Message Coexistence Mode

During the transition to SeND secured interfaces, network operators may want to run a particular interface with a mixture of nodes accepting secured and unsecured neighbor discovery messages. Perform this task to configure the coexistence mode for secure and nonsecure ND messages on the same interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 nd secured trustpoint** *trustpoint-name*
5. **no ipv6 nd secured full-secure**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type* *number* **Example:** `Device(config)# interface Ethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 4	**ipv6 nd secured trustpoint** *trustpoint-name* **Example:** `Device(config-if)# ipv6 nd secured trustpoint trustpoint1`	Enables SeND on an interface and specifies which trustpoint should be used.
Step 5	**no ipv6 nd secured full-secure** **Example:** `Device(config-if)# no ipv6 nd secured full-secure`	Provides the coexistence mode for secure and nonsecure ND messages on the same interface.

Customizing SeND Parameters

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 nd secured key-length** [[**minimum** | **maximum**] *value*]
4. **ipv6 nd secured sec-level minimum** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ipv6 nd secured key-length** [[**minimum** | **maximum**] *value*] **Example:** `Device(config)# ipv6 nd secured key-length minimum 512`	Configures the SeND key-length options.
Step 4	**ipv6 nd secured sec-level minimum** *value* **Example:** `Device(config)# ipv6 nd secured sec-level minimum 2`	Configures the minimum security level value that can be accepted from peers.

Configuring the SeND Time Stamp

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 nd secured timestamp** {**delta** *value* | **fuzz** *value*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface Ethernet 0/0`	Specifies an interface type and number, and places the device in interface configuration mode.

	Command or Action	Purpose

Step 4	**ipv6 nd secured timestamp** {**delta** *value* | **fuzz** *value*}
	
	Example:
	
	`Device(config-if)# ipv6 nd secured timestamp delta 600`
		Configures the SeND time stamp.

Configuration Examples for IPv6 Secure Neighbor Discovery

Example: Configuring Certificate Servers

```
crypto pki server CA
 issuer-name C=FR, ST=fr, L=example, O=cisco, OU=nsstg, CN=CA0  lifetime ca-certificate 700
 !
crypto pki trustpoint CA
 ip-extension prefix 2001::/16
 revocation-check crl
 rsakeypair CA
no shutdown
```

To display the certificate servers with IP extensions, use the **show crypto pki certificates verbose** command:

```
Device# show crypto pki certificates verbose

CA Certificate
  Status: Available
  Version: 3
  Certificate Serial Number (hex): 01
  Certificate Usage: Signature
  Issuer:
    c=FR
    st=fr
    l=example
    o=cisco
    ou=nsstg
    cn=CA0
  Subject:
    c=FR
    st=fr
    l=example
    o=cisco
    ou=nsstg
    cn=CA0
  Validity Date:
    start date: 09:50:52 GMT Feb 5 2009
    end   date: 09:50:52 GMT Jan 6 2011
  Subject Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
  Signature Algorithm: MD5 with RSA Encryption
  Fingerprint MD5: 87DB764F 29367A65 D05CEE3D C12E0AC3
  Fingerprint SHA1: 04A06602 86AA72E9 43F2DB33 4A7D40A2 E2ED3325
  X509v3 extensions:
    X509v3 Key Usage: 86000000
      Digital Signature
      Key Cert Sign
      CRL Signature
    X509v3 Subject Key ID: 75B477C6 B2CA7BBE C7866657 57C84A32 90CEFB5A
    X509v3 Basic Constraints:
        CA: TRUE
    X509v3 Authority Key ID: 75B477C6 B2CA7BBE C7866657 57C84A32 90CEFB5A
    Authority Info Access:
    X509v3 IP Extension:
        IPv6:
          2001::/16
  Associated Trustpoints: CA
```

Example: Configuring a Host to Enable SeND

```
 crypto key generate rsa label SEND modulus 1024
The name for the keys will be: SEND
% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
 ipv6 cga modifier rsakeypair SEND sec-level 1
 crypto pki trustpoint SEND
  enrollment url http://10.165.200.254
  revocation-check none
  exit
 crypto pki authenticate SEND
Certificate has the following attributes:
 Fingerprint MD5: FC99580D 0A280EB4 2EB9E72B 941E9BDA
 Fingerprint SHA1: 22B10EF0 9A519177 145EA4F6 73667837 3A154C53
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
 ipv6 nd secured sec-level minimum 1
 interface fastethernet 0/0
  ipv6 cga rsakeypair SEND
  ipv6 address FE80::260:3EFF:FE11:6770 link-local cga
  ipv6 nd secured trustanchor SEND
  ipv6 nd secured timestamp delta 300
  exit
 ipv6 nd secured full-secure
```

Use the **show running-config** command to verify the configuration:

```
Device# show running-config

Building configuration...
[snip]
crypto pki trustpoint SEND
 enrollment url http://10.165.200.225
 revocation-check none
!
interface Ethernet1/0
 ip address 10.165.202.129 255.255.255.0
 duplex half
 ipv6 cga rsakeypair SEND
 ipv6 address 2001:100::/64 cga
```

Example: Configuring a Device to Enable SeND

```
crypto key generate rsa label SEND modulus 1024
 ipv6 cga modifier rsakeypair SEND sec-level 1
 crypto pki trustpoint SEND
  subject-name C=FR, ST=PACA, L=Example, O=Cisco, OU=NSSTG, CN=device
  rsakeypair SEND
  revocation-check none
  exit
 crypto pki authenticate key1
Certificate has the following attributes:
 Fingerprint MD5: FC99580D 0A280EB4 2EB9E72B 941E9BDA
 Fingerprint SHA1: 22B10EF0 9A519177 145EA4F6 73667837 3A154C53
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
 crypto pki enroll SEND
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.
Password:
Re-enter password:
% The subject name in the certificate will include: C=FR, ST=fr, L=example, O=cisco, OU=nsstg,
 CN=device %
The subject name in the certificate will include: Device % Include the device serial number
 in the subject name? [yes/no]: no % Include an IP address in the subject name? [no]:
Request certificate from CA? [yes/no]: yes % Certificate request sent to Certificate
Authority % The 'show crypto pki certificate SEND verbose' commandwill show the fingerprint.
*Feb  5 09:40:37.171: CRYPTO_PKI:  Certificate Request Fingerprint MD5:
A6892F9F 23561949 4CE96BB8 CBC85 E64
*Feb  5 09:40:37.175: CRYPTO_PKI:  Certificate Request Fingerprint SHA1:
30832A66 E6EB37DF E578911D 383F 96A0 B30152E7
*Feb  5 09:40:39.843: %PKI-6-CERTRET: Certificate received from Certificate Authority
 interface fastethernet 0/0
  ipv6 nd secured sec-level minimum 1
  ipv6 cga rsakeypair SEND
  ipv6 address fe80:: link-local cga
  ipv6 nd secured trustanchor SEND
  ipv6 nd secured timestamp delta 300
  exit
 ipv6 nd secured full-secure
```

To verify that the certificates are generated, use the **show crypto pki certificates** command:

```
Device# show crypto pki certificates

Certificate
  Status: Available
  Certificate Serial Number: 0x15
  Certificate Usage: General Purpose
  Issuer:
    cn=CA
  Subject:
    Name: Device
    hostname=Device
    c=FR
    st=fr
    l=example
    o=cisco
    ou=nsstg
    cn=device
  Validity Date:
    start date: 09:40:38 UTC Feb 5 2009
    end   date: 09:40:38 UTC Feb 5 2010
  Associated Trustpoints: SEND
CA Certificate
  Status: Available
  Certificate Serial Number: 0x1
  Certificate Usage: Signature
  Issuer:
    cn=CA
  Subject:
    cn=CA
  Validity Date:
    start date: 10:54:53 UTC Jun 20 2008
    end   date: 10:54:53 UTC Jun 20 2011
  Associated Trustpoints: SEND
```

To verify the configuration, use the **show running-config** command:

```
Device# show running-config

Building configuration...
```

```
[snip]
crypto pki trustpoint SEND
 enrollment url http://209.165.201.1
 subject-name C=FR, ST=fr, L=example, O=cisco, OU=nsstg, CN=device
 revocation-check none  rsakeypair SEND !
interface Ethernet1/0
 ip address 209.165.200.225 255.255.255.0
 duplex half
 ipv6 cga rsakeypair SEND
 ipv6 address FE80:: link-local cga
 ipv6 address 2001:100::/64 cga
```

Example: Configuring a SeND Trustpoint

```
crypto key generate rsa label SEND
  Choose the size of the key modulus in the range of 360 to 2048 for your
 General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.
How many bits in the modulus [512]: 778
% Generating 778 bit RSA keys, keys will be non-exportable...[OK]
 ipv6 cga modifier rsakeypair SEND sec-level 1
 crypto pki trustpoint trustpoint1
  subject-name C=FR, ST=fr, L=example, O=cisco, OU=nsstg, CN=sa14-72b
  rsakeypair SEND
  enrollment terminal
  ip-extension unicast prefix 2001:100:1://48
  exit
 crypto pki authenticate trustpoint1
 crypto pki enroll trustpoint1
 crypto pki import trustpoint1 certificate
 interface Ethernet 0/0
  ipv6 nd secured trustpoint trustpoint1
```

Example: Configuring SeND Trust Anchors

```
! Configure the location of the CS we trust !
 crypto pki trustpoint B1
  enrollment terminal
  crypto pki authenticate anchor1
  exit
! Only Query a certificate signed by the CS at B2 on this interface !
 interface Ethernet 0/0
  ip address 204.209.1.54 255.255.255.0
  ipv6 cga rsakeypair SEND
  ipv6 address 2001:100::/64 cga
  ipv6 nd secured trustanchor anchor1
```

Example: Configuring CGA Address Generation on an Interface

```
enable
 configure terminal
 interface fastEthernet 0/0
  ipv6 cga rsakeypair SEND
  ipv6 address 2001:100::/64 cga
  exit
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	*Cisco IOS IPv6 Command Reference*
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for IPv6 Secure Neighbor Discovery

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 22: Feature Information for IPv6 Secure Neighbor Discovery

Feature Name	Releases	Feature Information
IPv6 Secure Neighbor Discovery	12.4(24)T	The SeND protocol is designed to counter the threats of the ND protocol. SeND defines a set of neighbor discovery options and two neighbor discovery messages. SeND also defines a new autoconfiguration mechanism to establish address ownership. The following commands were introduced or modified: **auto-enroll**, **crypto key generate rsa**, **crypto pki authenticate**, **crypto pki enroll**, **crypto pki import**, **enrollment terminal (ca-trustpoint)**, **enrollment url (ca-trustpoint)**, **fingerprint**, **ip-extension**, **ip http server**, **ipv6 address**, **ipv6 address link-local**, **ipv6 cga modifier rsakeypair**, **ipv6 cga modifier rsakeypair (interface)**, **ipv6 nd secured certificate-db**, **ipv6 nd secured full-secure**, **ipv6 nd secured full-secure (interface)**, **ipv6 nd secured key-length**, **ipv6 nd secured sec-level**, **ipv6 nd secured timestamp**, **ipv6 nd secured timestamp-db**, **ipv6 nd secured trustanchor**, **ipv6 nd secured trustpoint**, **password (ca-trustpoint)**, **revocation-check**, **rsakeypair**, **serial-number (ca-trustpoint)**, **show ipv6 cga address-db**, **show ipv6 cga modifier-db**, **show ipv6 nd secured certificates**, **show ipv6 nd secured counters interface**, **show ipv6 nd secured nonce-db**, **show ipv6 nd secured timestamp-db**, **subject-name**.

Glossary

- **CA**—certification authority.

- **CGA—**cryptographically generated address.
- **CPA—**certificate path answer.
- **CPR—**certificate path response.
- **CPS—**certification path solicitation. The solicitation message used in the addressing process.
- **CRL—**certificate revocation list.
- **CS—**certification server.
- **CSR—**certificate signing request.
- **DAD—**duplicate address detection. A mechanism that ensures two IPv6 nodes on the same link are not using the same address.
- **DER—**distinguished encoding rules. An encoding scheme for data values.
- **nonce—**An unpredictable random or pseudorandom number generated by a node and used once. In SeND, nonces are used to ensure that a particular advertisement is linked to the solicitation that triggered it.
- **non-SeND node—**An IPv6 node that does not implement SeND but uses only the Neighbor Discovery Protocol without security.
- **NUD—**neighbor unreachability detection. A mechanism used for tracking neighbor reachability.
- **PACL—**port-based access list.
- **PKI—**public key infrastructure.
- **RA—**router advertisement.
- **RD—**Router discovery allows the hosts to discover what devices exist on the link and what subnet prefixes are available. Router discovery is a part of the Neighbor Discovery Protocol.
- **Router Authorization Certificate—**A public key certificate.
- **SeND node—**An IPv6 node that implements SeND.
- **trust anchor—**An entity that the host trusts to authorize devices to act as devices. Hosts are configured with a set of trust anchors to protect device discovery.