CHAPTER 42

Understanding Cisco IOS ACL Support

This chapter describes Cisco IOS access control list (ACL) support in Cisco IOS Software Release 12.2SX:

For complete information about configuring Cisco IOS ACLs, see the *Cisco IOS Security Configuration Guide*, Release 12.2, "Traffic Filtering and Firewalls," at this URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/ftrafwl/index.htm

ACL Support in Hardware and Software

ACLs can be processed in hardware by the Policy Feature Card (PFC), a Distributed Forwarding Card (DFC), or in software by the route processor (RP):

- ACL flows that match a "deny" statement in standard and extended ACLs (input and output) are dropped in hardware if "ip unreachables" is disabled.
- ACL flows that match a "permit" statement in standard and extended ACLs (input and output) are processed in hardware.
- VLAN ACL (VACL) and port ACL (PACL) flows are processed in hardware. If a field that is specified in a VACL or PACL is not supported by hardware processing, then that field is ignored (for example, the **log** keyword in an ACL), or the whole configuration is rejected (for example, a VACL containing IPX ACL parameters).
- VACL logging is processed in software.
- Dynamic ACL flows are processed in hardware.
- Idle timeout is processed in software.

Note Idle timeout is not configurable. Cisco IOS Software Release 12.2SX does not support the **access-enable host timeout** command.

- IP accounting for an ACL access violation on a given port is supported by forwarding all denied packets for that port to the RP for software processing without impacting other flows.
- The PFC does not provide hardware support for Cisco IOS IPX ACLs. Cisco IOS IPX ACLs are supported in software on the RP.
- Extended name-based MAC address ACLs are supported in hardware.
- The following ACL types are processed in software:
  - Internetwork Packet Exchange (IPX) access lists
  - Standard XNS access list
  - Extended XNS access list
  - DECnet access list
  - Extended MAC address access list
  - Protocol type-code access list

Note IP packets with a header length of less than five will not be access controlled.

- Unless you configure optimized ACL logging (OAL), flows that require logging are processed in software without impacting nonlogged flow processing in hardware (see the "Optimized ACL Logging" section on page 42-7).
- The forwarding rate for software-processed flows is substantially less than for hardware-processed flows.
- When you enter the **show ip access-list** command, the match count displayed does not include packets processed in hardware.

Cisco IOS ACL Configuration Guidelines and Restrictions

The following guidelines and restrictions apply to Cisco IOS ACLs configured for use with any feature:

- You can apply Cisco IOS ACLs directly to Layer 3 ports and to VLAN interfaces.
- You can apply VLAN ACLs and port ACLs to Layer 2 interfaces (see Chapter 43, "Configuring Port ACLs and VLAN ACLs").
- Each type of ACL (IP, IPX, and MAC) filters only traffic of the corresponding type. A Cisco IOS MAC ACL never matches IP or IPX traffic.
- The PFC does not provide hardware support for Cisco IOS IPX ACLs. Cisco IOS IPX ACLs are supported in software on the route processor (RP).
- By default, the RP sends Internet Control Message Protocol (ICMP) unreachable messages when a packet is denied by an access group.

  With the **ip unreachables** command enabled (which is the default), the switch processor (SP) drops most of the denied packets in hardware and sends only a small number of packets to the RP to be dropped (10 packets per second, maximum), which generates ICMP-unreachable messages.

  To eliminate the load imposed on the RP CPU by the task of dropping denied packets and generating ICMP-unreachable messages, you can enter the **no ip unreachables** interface configuration command to disable ICMP unreachable messages, which allows all access group-denied packets to be dropped in hardware.

- ICMP unreachable messages are not sent if a packet is denied by a VACL or a PACL.
- Use named ACLs (instead of numbered ACLs) because this causes less CPU usage when creating or modifying ACL configurations and during system restarts. When you create ACL entries (or modify existing ACL entries), the software performs a CPU-intensive operation called an ACL merge to load the ACL configurations into the PFC hardware. An ACL merge also occurs when the startup configuration is applied during a system restart.

  With named ACLs, the ACL merge is triggered only when the user exits the **named-acl** configuration mode. However, with numbered ACLs, the ACL merge is triggered for every ACE definition and results in a number of intermediate merges during ACL configuration.

Policy-Based ACLs

Release 12.2(33)SXH and later releases support policy-based ACLs (PBACLs). The following sections describe PBACLs:

- Understanding PBACLs, page 42-3
- PBACL Guidelines and Restrictions, page 42-3
- Configuring PBACL, page 42-4

Understanding PBACLs

PBACLs provide the capability to apply access control policies across object groups. An object group is a group of users or servers.

You define an object group as a group of IP addresses or as a group of protocol ports. You then create access control entries (ACEs) that apply a policy (such as permit or deny) to the object group. For example, a policy-based ACE can permit a group of users to access a group of servers.

An ACE defined using a group name is equivalent to multiple ACEs (one applied to each entry in the object group). The system expands the PBACL ACE into multiple Cisco IOS ACEs (one ACE for each entry in the group) and populates the ACEs in the TCAM. Therefore, the PBACL feature reduces the number of entries you need to configure but does not reduce TCAM usage.

If you make changes in group membership, or change the contents of an ACE that uses an access group, the system updates the ACEs in the TCAM. The following types of changes trigger the update:

- Adding a member to a group
- Deleting a member from a group
- Modifying the policy statements in an ACE that uses an access group

You configure a PBACL using extended Cisco IOS ACL configuration commands. As with regular ACEs, you can associate the same access policy with one or more interfaces.

When you configure an ACE, you can use an object group to define the source, the destination, or both.

PBACL Guidelines and Restrictions

When configuring PBACLs, note the following guidelines and restrictions:

- PBACLs are supported on Layer 3 interfaces (such as routed interfaces and VLAN interfaces).
- The PBACL feature only supports IPv4 ACEs.

- The PBACL feature is supported for Cisco IOS ACLs, but is not supported on VLAN ACLs or port ACLs. The keywords **reflexive** and **evaluate** are not supported.
- Feature interactions for policy-based ACLs are the same as for Cisco IOS ACLs.

Configuring PBACL

Configure PBACLs using the following tasks:

Configuring a PBACL IP Address Object Group

To create or modify a PBACL IP address object group, perform this task:

	Command	Purpose
Step 1	`Router(config)#` **`object-group ip address`** *`object_group_name`*	Defines object group name and enters IP-address object-group configuration mode.
	`Router(config)#` **`no object-group ip address`** *`object_group_name`*	Deletes the object group with the specified name.
Step 2	`Router(config-ipaddr-ogroup)# {`*`ip_address mask`*`}	{`**`host`**` {`*`name`*`	`*`ip_address`*`} }`	Configures a member of the group. The member is either a network address plus mask or a host (identified by host name or IP address).
	`Router(config-ipaddr-ogroup)#` **`no`** `{`*`ip_address mask`*`}	{`**`host`**` {`*`name`*`	`*`ip_address`*`} }`	Deletes a member of the group.
Step 3	`Router(config-ipaddr-ogroup)# {`**`end`**`}	{`**`exit`**`}`	To leave the configuration mode, enter the **end** command. To leave the IP-address object-group configuration mode, enter the **exit** command.
Step 4	`Router#` **`show object-group`** `[`*`object_group_name`*`]`	Displays the object-group configuration for the named group (or for all groups if no name is entered).

The following example creates an object group with three hosts and a network address:

```
Router(config)# object-group ip address myAG
Router(config-ipaddr-pgroup)# host 10.20.20.1
Router(config-ipaddr-pgroup)# host 10.20.20.5
Router(config-ipaddr-pgroup)# 10.30.0.0 255.255.0.0
```

Configuring a PBACL Protocol Port Object Group

To create or modify a PBACL protocol port object group, perform this task:

	Command	Purpose
Step 1	`Router(config)#` **`object-group ip port`** *`object_group_name`*	Defines object group name and enters port object-group configuration mode.
	`Router(config)#` **`no object-group ip port`** *`object_group_name`*	Deletes the object group with the specified name.
Step 2	`Router(config-port-ogroup)# {`**`eq`** *`number`*`} | {`**`gt`** *`number`*`} | {`**`lt`** *`number`*`} | {`**`neq`** *`number`*`} | {`**`range`** *`number number`*`}`	Configures a member of the group. The member is either equal to or not equal to a port number, less than or greater than a port number, or a range of port numbers.
	`Router(config-port-ogroup)#` **`no`** `{ {`**`eq`** *`number`*`} | {`**`gt`** *`number`*`} | {`**`lt`** *`number`*`} | {`**`neq`** *`number`*`} | {`**`range`** *`number number`*`} }`	Deletes a member of the group.
Step 3	`Router(config-port-ogroup)#` **`end`** `|` **`exit`**	To leave the configuration mode, enter the **end** command. To leave the port object-group configuration mode, enter the **exit** command.
Step 4	`Router#` **`show object-group`** `[`*`object_group_name`*`]`	Displays the object-group configuration for the named group (or for all groups if no name is entered).

The following example creates a port object group that matches protocol port 100 and any port greater than 200, except 300:

```
Router(config)# object-group ip port myPG
Router(config-port-pgroup)# eq 100
Router(config-port-pgroup)# gt 200
Router(config-port-pgroup)# neq 300
```

Configuring ACLs with PBACL Object Groups

To configure an ACL to use a PBACL object group, perform this task:

	Command	Purpose
Step 1	`Router(config)#` **`ip access-list extended`** *`acl_name`*	Defines an extended ACL with the specified name and enters extended-ACL configuration mode.
	`Router(config)#` **`no ip access-list extended`** *`acl_name`*	Deletes the ACL with the specified name.
Step 2	`Router(config-ext-nacl)#` **`permit tcp addrgroup`** *`object_group_name`* **`addrgroup`** *`object_group_name`*	Configures an ACE for TCP traffic using IP address object group as the source policy and an object group as the destination policy.
	`Router(config-ext-nacl)#` **`no ip addrgroup`** *`object_group_name`* **`addrgroup`** *`object_group_name`*	Deletes an entry in the extended ACL.
Step 3	`Router(config-ext-nacl)#` **`exit`**	Exits extended ACL configuration mode.
Step 4	`Router#` **`show access-lists`** `[`*`acl_name`*`]`	Displays the object-group configuration for the named group (or for all groups if no name is entered).

The following example creates an access list that permits packets from the users in myAG if the protocol ports match the ports specified in myPG:

```
Router(config)# ip access-list extended my-pbacl-policy
Router(config-ext-nacl)# permit tcp addrgroup myAG portgroup myPG any
Router(config-ext-nacl)# deny tcp any any
Router(config-ext-nacl)# exit
Router(config)# exit
Router# show ip access-list my-pbacl-policy
Extended IP access list my-pbacl-policy
10  permit tcp addrgroup AG portgroup PG any
20  permit tcp any any

Router# show ip access-list my-pbacl-policy expand
Extended IP access list my-pbacl-policy expanded
20 permit tcp host 10.20.20.1 eq 100 any
20 permit tcp host 10.20.20.1 gt 200 any
20 permit tcp host 10.20.20.1 neq 300 any
20 permit tcp host 10.20.20.5 eq 100 any
20 permit tcp host 10.20.20.5 gt 200 any
20 permit tcp host 10.20.20.5 neq 300 any
20 permit tcp 10.30.0.0 255.255.0.0 eq 100 any
20 permit tcp 10.30.0.0 255.255.0.0 gt 200 any
20 permit tcp 10.30.0.0 255.255.0.0 neq 300 any
```

Configuring PBACL on an Interface

To configure a PBACL on an interface, use the **ip access-group** command. The command syntax and usage is the same as for Cisco IOS ACLs. For additional information, see the "Cisco IOS ACL Configuration Guidelines and Restrictions" section on page 42-2.

The following example shows how to associate access list my-pbacl-policy with VLAN 100:

```
Router(config)# int vlan 100
Router(config-if)# ip access-group mp-pbacl-policy in
```

Configuring IPv6 Address Compression

ACLs are implemented in hardware in the Policy Feature Card (PFC), which uses the source or destination IP address and port number in the packet to index the ACL table. The index has a maximum address length of 128 bits.

The IP address field in an IPv6 packet is 128 bits, and the port field is 16 bits. To use full IPv6 addresses in the ACL hardware table, you can turn on compression of IPv6 addresses using the **mls ipv6 acl compress address unicast** command. This feature compresses the IPv6 address (including port) into 128 bits by removing 16 unused bits from the IPv6 address. Compressible address types can be compressed without losing any information. See Table 42-1 for details about the compression methods.

By default, the command is set for no compression.

Caution Do not enable the compression mode if you have noncompressible address types in your network. A list of compressible address types and the address compression method are listed in Table 42-1.

Table 42-1 Compressible Address Types and Methods

Address Type	Compression Method
EUI-64 based on MAC address	This address is compressed by removing 16 bits from bit locations [39:24]. No information is lost when the hardware compresses these addresses.
Embedded IPv4 address	This address is compressed by removing the upper 16 bits. No information is lost when the hardware compresses these addresses.
Link Local	These addresses are compressed by removing the zeros in bits [95:80] and are identified using the same packet type as the embedded IPv4 address. No information is lost when the hardware compresses these addresses.
Others	If the IPv6 address does not fall into any of the above categories, it is classified as other. If the IPv6 address is classified as other, the following occurs: • If the compress mode is on, the IPv6 address is compressed similarly to the EUI-64 compression method (removal of bits [39:24]) to allow for the Layer 4 port information to be used as part of the key used to look up the QoS TCAM, but Layer 3 information is lost. • If the global compression mode is off, the entire 128 bits of the IPv6 address are used. The Layer 4 port information cannot be included in the key to look up the QoS TCAM because of the size constraints on the IPv6 lookup key.

To turn on the compression of IPv6 addresses, enter the **mls ipv6 acl compress address unicast** command. To turn off the compression of IPv6 addresses, enter the **no** form of this command.

This example shows how to turn on address compression for IPv6 addresses:

```
Router(config)# mls ipv6 acl compress address unicast
Router(config)#
```

This example shows how to turn off address compression for IPv6 addresses:

```
Router(config)# no mls ipv6 acl compress address unicast
Router(config)#
```

Optimized ACL Logging

These sections describe optimized ACL logging (OAL):

- Understanding OAL, page 42-8
- OAL Guidelines and Restrictions, page 42-8
- Configuring OAL, page 42-8

Understanding OAL

OAL provides hardware support for ACL logging. Unless you configure OAL, packets that require logging are processed completely in software on the RP. OAL permits or drops packets in hardware on the PFC3 and uses an optimized routine to send information to the RP to generate the logging messages.

OAL Guidelines and Restrictions

The following guidelines and restrictions apply to OAL:

- OAL and VACL capture are incompatible. Do not configure both features on the switch. With OAL configured, use SPAN to capture traffic.
- OAL is supported only on the PFC3.
- OAL supports only IPv4 unicast packets.
- OAL supports VACL and PACL logging of permitted ingress traffic.
- OAL does not provide hardware support for the following:
  - Reflexive ACLs
  - ACLs used to filter traffic for other features (for example, QoS)
  - ACLs for unicast reverse path forwarding (uRPF) check exceptions
  - Exception packets (for example, TTL failure and MTU failure)
  - Packets with IP options
  - Packets addressed at Layer 3 to the router
  - Packets sent to the RP to generate ICMP unreachable messages
  - Packets being processed by features not accelerated in hardware
- To provide OAL support for denied packets, enter the **mls rate-limit unicast ip icmp unreachable acl-drop 0** command.

Configuring OAL

These sections describe how to configure OAL:

- Configuring OAL Global Parameters, page 42-9
- Configuring OAL on an Interface, page 42-9
- Displaying OAL Information, page 42-10
- Clearing Cached OAL Entries, page 42-10

Note

- For complete syntax and usage information for the commands used in this section, see the Cisco IOS Software Releases 12.2SX Command References.
- To provide OAL support for denied packets, enter the **mls rate-limit unicast ip icmp unreachable acl-drop 0** command.

Configuring OAL Global Parameters

To configure global OAL parameters, perform this task:

Command	Purpose
`Router(config)#` **`logging ip access-list cache`** `{{`**`entries`** *`number_of_entries`*`} | {`**`interval`** *`seconds`*`} | {`**`rate-limit`** *`number_of_packets`*`} | {`**`threshold`** *`number_of_packets`*`}}`	Sets OAL global parameters.
`Router(config)#` **`no logging ip access-list cache`** `{`**`entries`** `|` **`interval`** `|` **`rate-limit`** `|` **`threshold`**`}`	Reverts OAL global parameters to defaults.

When configuring OAL global parameters, note the following information:

- **entries** *number_of_entries*
  - Sets the maximum number of entries cached.
  - Range: 0–1,048,576 (entered without commas).
  - Default: 8000.
- **interval** *seconds*
  - Sets the maximum time interval before an entry is sent to be logged. Also if the entry is inactive for this duration it is removed from the cache.
  - Range: 5–86,400 (1440 minutes or 24 hours, entered without commas).
  - Default: 300 seconds (5 minutes).
- **rate-limit** *number_of_packets*
  - Sets the number of packets logged per second in software.
  - Range: 10–1,000,000 (entered without commas).
  - Default: 0 (rate limiting is off and all packets are logged).
- **threshold** *number_of_packets*
  - Sets the number of packet matches before an entry is logged.
  - Range: 1–1,000,000 (entered without commas).
  - Default: 0 (logging is not triggered by the number of packet matches).

Configuring OAL on an Interface

To configure OAL on an interface, perform this task:

	Command	Purpose
Step 1	`Router(config)#` **`interface`** `{{`*`type`*[1] *`slot/port`*`}`	Specifies the interface to configure.
Step 2	`Router(config-if)#` **`logging ip access-list cache in`**	Enables OAL for ingress traffic on the interface.
	`Router(config-if)#` **`no logging ip access-list cache`**	Disables OAL on the interface.
Step 3	`Router(config-if)#` **`logging ip access-list cache out`**	Enables OAL for egress traffic on the interface.
	`Router(config-if)#` **`no logging ip access-list cache`**	Disables OAL on the interface.

1. *type* = any that supports Layer 3-switched traffic.

Displaying OAL Information

To display OAL information, perform this task:

Command	Purpose
`Router # show logging ip access-list cache`	Displays OAL information.

Clearing Cached OAL Entries

To clear cached OAL entries, perform this task:

Command	Purpose
`Router # clear logging ip access-list cache`	Clears cached OAL entries.

Guidelines and Restrictions for Using Layer 4 Operators in ACLs

These sections describe guidelines and restrictions when configuring ACLs that include Layer 4 port operations:

- Determining Layer 4 Operation Usage, page 42-10
- Determining Logical Operation Unit Usage, page 42-11

Determining Layer 4 Operation Usage

You can specify these types of operations:

- gt (greater than)
- lt (less than)
- neq (not equal)
- eq (equal)
- range (inclusive range)

We recommend that you do not specify more than *nine different* operations on the same ACL. If you exceed this number, each new operation might cause the affected ACE to be translated into more than one ACE.

Use the following two guidelines to determine Layer 4 operation usage:

- Layer 4 operations are considered different if the operator or the operand differ. For example, in this ACL there are three different Layer 4 operations (“gt 10” and “gt 11” are considered two different Layer 4 operations):

```
... gt 10 permit
... lt 9 deny
... gt 11 deny
```

Note There is no limit to the use of “eq” operators as the “eq” operator does not use a logical operator unit (LOU) or a Layer 4 operation bit. See the “Determining Logical Operation Unit Usage” section on page 42-11 for a description of LOUs.

- Layer 4 operations are considered different if the same operator/operand couple applies once to a source port and once to a destination port. For example, in this ACL there are two different Layer 4 operations because one ACE applies to the source port and one applies to the destination port.

```
... Src gt 10 ...
... Dst gt 10
```

Determining Logical Operation Unit Usage

Logical operation units (LOUs) are registers that store operator-operand couples. All ACLs use LOUs. There can be up to 32 LOUs; each LOU can store two different operator-operand couples with the exception of the range operator. LOU usage per Layer 4 operation is as follows:

- gt uses 1/2 LOU
- lt uses 1/2 LOU
- neq uses 1/2 LOU
- range uses 1 LOU
- eq does not require a LOU

For example, this ACL would use a single LOU to store two different operator-operand couples:

```
... Src gt 10 ...
... Dst gt 10
```

A more detailed example follows:

```
ACL1
... (dst port) gt 10 permit
... (dst port) lt 9 deny
... (dst port) gt 11 deny
... (dst port) neq 6 permit
... (src port) neq 6 deny
... (dst port) gt 10 deny

ACL2
... (dst port) gt 20 deny
... (src port) lt 9 deny
... (src port) range 11 13 deny
... (dst port) neq 6 permit
```

The Layer 4 operations and LOU usage is as follows:

- ACL1 Layer 4 operations: 5
- ACL2 Layer 4 operations: 4
- LOUs: 4

An explanation of the LOU usage follows:

- LOU 1 stores “gt 10” and “lt 9”
- LOU 2 stores “gt 11” and “neq 6”

- LOU 3 stores “gt 20” (with space for one more)
- LOU 4 stores “range 11 13” (range needs the entire LOU)