Object Groups for ACLs

First Published: July 11, 2008
Last Updated: September 6, 2010

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 feature information and caveats, see 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 for Object Groups for ACLs" section on page 17.

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

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

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.

To configure the Object Groups for ACLs feature, you should understand the following concepts:

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 ACE could allow a group of users to have access only to a specific group of servers. In an object group-based ACL, you can create a single ACE that uses an object group name instead of creating many ACEs (which would require each one 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 to a server 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 could control its group membership, and the administrator could own the ACE itself to control which departments can contact one another.

You can use object groups as members (children) of other object groups. For example, you can create an ENG-ALL address group that contains the ENG-EAST and ENG-WEST address groups. You can use an unlimited number of levels of nested (child) object groups (however, a maximum of two levels is recommended).

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. These object groups can be used 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
- Ranges of IP addresses
- Subnets

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))
- ICMP types (such as echo, echo-reply, or host-unreachable)
- Top-level protocols (such as TCP, User Datagram Protocol (UDP), or Encapsulating Security Payload (ESP))
- Other service object groups

ACLs Based on Object Groups

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

You can apply object group-based ACLs to interfaces that are configured in a VPN routing and forwarding (VRF) instance or features that are used within a VRF context.

You can add to, delete from, or change objects in an object group membership list dynamically (meaning without deleting and redefining the object group). Also, you can add to, delete from, or change objects in an object group membership list without redefining the ACL ACE that is using the object group (meaning changing the object group without deleting the ACE and then redefining the ACE after the change). You can add objects to groups, delete them from groups, and then ensure that the changes are properly 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 source and destination groups.

You cannot delete an object group that is being used within an ACL or a CPL policy.

How to Configure Object Group-Based ACLs

To configure the Object Groups for ACLs feature, you first create one or more object groups. These can be any combination of network object groups (containing 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 ACEs that apply a policy (such as **permit** or **deny**) to those object groups.

This section contains the following procedures:

- Creating a Network Object Group, page 4 (optional)
- Creating a Service Object Group, page 6 (optional)
- Creating an Object Group-Based ACL, page 8 (required)
- Applying an Object Group-Based ACL to an Interface, page 11 (required)
- Verifying Object Groups for ACLs, page 12 (optional)

Creating a Network Object Group

A network object group containing a single object (such as a single IP address, a hostname, another network object group, or a subnet) or multiple objects (such as a combination of multiple IP addresses, hostnames, a range of IP addresses, other object network groups, or subnets), can be used with an ACL in 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. **range** *host-address1 host-address2*
8. **any**
9. **group-object** *nested-object-group-name*
10. Repeat some combination of Steps 5 through 9 until you have specified the objects on which you want to base your object group.
11. **end**

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	**object-group network** *object-group-name* **Example:** `Router(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:** `Router(config-network-group)# description test engineers`	(Optional) Specifies a description of the object group. • You can use up to 200 characters.
Step 5	**host** {*host-address* | *host-name*} **Example:** `Router(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:** `Router(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	**range** *host-address1 host-address2* **Example:** `Router(config-network-group)# range 209.165.200.242 209.165.200.243`	(Optional) Specifies a range of host IP addresses. • If you specify a range of 000.000.000.000 to 255.255.255.255, the effect is the same as the use of the **any** command. • If you specify the same IP address for the *host-address1* and *host-address2* arguments, the effect is the same as the use of the **host** command—the identical IP address specified becomes the single host IP address for the object group.
Step 8	**any** **Example:** `Router(config-network-group)# any`	(Optional) Specifies any host IP address in the range 0.0.0.0 to 255.255.255.255.

	Command or Action	Purpose
Step 9	**group-object** *nested-object-group-name* **Example:** `Router(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 10	Repeat some combination of Steps 5 through 9 until you have specified the objects on which you want to base your object group.	—
Step 11	**end** **Example:** `Router(config-network-group)# end`	Returns to privileged EXEC mode.

Creating a Service Object Group

You can use a service object group to specify specific TCP and/or UDP ports or ranges of them. When the service object group is associated with an ACL, this service object group-based ACL can be used to control access to the 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 some combination of Steps 5 through 8 until you have specified the objects on which you want to base your object group.

10. **end**

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	**object-group service** *object-group-name* **Example:** `Router(config)# object-group service my_service_object_group`	Defines the object group name and enters service object-group configuration mode.
Step 4	**description** *description-text* **Example:** `Router(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:** `Router(config-service-group)# ahp`	(Optional) Specifies an IP protocol number or name.
Step 6	**tcp** | **udp** | **tcp-udp** [**source** {{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*}] [{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*] **Example:** `Router(config-service-group)# tcp-udp range 2000 2005`	(Optional) Specifies TCP, UDP, or both.
Step 7	**icmp** *icmp-type* **Example:** `Router(config-service-group)# icmp conversion-error`	(Optional) Specifies the decimal number or name of an ICMP type.

	Command or Action	Purpose
Step 8	**group-object** *nested-object-group-name* **Example:** `Router(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 some combination of Steps 5 through 8 until you have specified the objects on which you want to base your object group.	—
Step 10	**end** **Example:** `Router(config-service-group)# end`	Returns to privileged EXEC mode.

Creating an Object Group-Based ACL

When creating an object group-based ACL, you 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 ACEs that reference object groups within the same object group-based ACL. Also, you can 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 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**

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** *access-list-name* **Example:** `Router(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:** `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 Marketing 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 209.165.200.244 255.255.255.224 host 209.165.200.245 log`	(Optional) Denies any packet that matches all of the conditions specified in the statement. • 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 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:** `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.

	Command or Action	Purpose
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. • 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 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`	Returns to privileged EXEC mode.

Applying an Object Group-Based ACL to an Interface

You use the **ip access-group** command to apply an object group-based ACL to an interface. The command syntax and usage are the same as for conventional ACLs. The object group-based 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:** `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 vlan 100`	Specifies the interface type and number and enters interface configuration mode.
Step 4	**ip access-group** {*access-list-name* | *access-list-number*} {**in** | **out**} **Example:** `Router(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:** `Router(config-if)# end`	Returns to privileged EXEC mode.

Verifying Object Groups for ACLs

Perform this task to verify 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:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show object-group** [*object-group-name*] **Example:** `Router# 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:** `Router# 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, a range of IP addresses, and a subnet as objects:

```
Router> enable
Router# configure terminal
Router(config)# object-group network my_network_object_group
Router(config-network-group)# host 209.165.200.237
Router(config-network-group)# host 209.165.200.238
Router(config-network-group)# range 209.165.200.239 209.165.200.240
Router(config-network-group)# 209.165.200.241 255.255.255.224
```

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

```
Router> enable
Router# configure terminal
Router(config)#object-group network sjc_ftp_servers
Router(config-network-group)# host sjc.eng.ftp
Router(config-network-group)# host 209.165.200.242
Router(config-network-group)# 209.165.200.225 255.255.255.224
Router(config-network-group)# group-object sjc_eng_ftp_servers
```

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 (child) named sjc_eng_svcs as objects:

```
Router> enable
Router# configure terminal
Router(config)# object-group service my_service_object_group
Router(config-service-group)# icmp echo
Router(config-service-group)# tcp smtp
Router(config-service-group)# tcp telnet
Router(config-service-group)# tcp source range 1 65535 telnet
Router(config-service-group)# udp domain
Router(config-service-group)# tcp-udp range 2000 2005
Router(config-service-group)# group-object sjc_eng_svcs
```

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:

```
Router> enable
Router# configure terminal
Router(config)# ip access-list extended my_ogacl_policy
Router(config-ext-nacl)# permit object-group my_service_object_group object-group
my_network_object_group any
Router(config-ext-nacl)# deny tcp any any
Router(config-ext-nacl)# exit
Router(config)# exit
```

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:

```
Router> enable
Router# configure terminal
Router(config)# interface vlan 100
Router(config-if)# ip access-group my_ogacl_policy in
Router(config-if)# end
```

Example: Verifying Object Groups for ACLs

The following example shows how to display all object groups:

```
Router> enable
Router# 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:

```
Router# show ip access-list my_ogacl_policy

Extended IP access list my_ogacl_policy
10  permit object-group eng_service any any
```

Additional References

Related Documents

Related Topic	Document Title
General information about ACLs	*IP Access List Overview*
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Security commands	*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
No new or modified RFCs are supported by this feature, and support for existing RFCs 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 Object Groups for ACLs

Table 1 lists the features in this module and provides links to specific configuration information.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 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.

Table 1 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 sections provide information about this feature: • ACLs Based on Object Groups, page 3 • Object Groups, page 2 • Creating a Network Object Group, page 4 • Creating a Service Object Group, page 6 • Creating an Object Group-Based ACL, page 8 • Applying an Object Group-Based ACL to an Interface, page 11 • Verifying Object Groups for ACLs, page 12 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**.