6

Access Control Lists and Traffic Control

Terms you'll need to understand:

✓ Access list
✓ Access group
✓ Turbo ACLs
✓ Object grouping
✓ Allowing traffic in
✓ Controlling traffic out
✓ Conduits

Techniques you'll need to master:

✓ Access lists
✓ Turbo ACLs
✓ Network object groups
✓ Protocol object groups
✓ Nesting object groups
✓ ICMP through the PIX

The Cisco PIX firewall has several commands in its arsenal to control traffic flow: conduit commands to control traffic from lower security level interfaces to higher security level interfaces; outbound filter commands to control traffic from higher security level interfaces to lower level interfaces; and Cisco Access Control List (ACL) commands. The ACL commands can do everything conduit and outbound filter commands do, but they do them better and even come with powerful features such as turbo ACLs and object grouping. This chapter talks about the uses of conduits, outbound filters, ACLs, turbo ACLs, and object grouping.

Controlling Traffic Coming In

The previous chapter introduced the use of the `conduit` command, which allows you to enable traffic initiated from lower security level interfaces to pass through to higher security level interfaces. Originally, the PIX firewall started with the `conduit` command, but Cisco has introduced the IOS-style ACL into newer PIX images. Cisco recommends doing away with the older `conduit` commands and prefers the newer ACL commands in their place. Before we jump into the recommended ACL commands for letting traffic in, however, let's explore the `conduit` command and some of its features.

The conduit Command

The `conduit` command makes an exception in the ASA to permit or deny specific traffic from lower security level interfaces to pass to higher security level interfaces. Figure 6.1 displays the traffic flow from the outside to the inside interface. The traffic first enters the PIX via the global IP address assigned to the outside interface. Then the conduit entries are checked to verify whether a permit match exists. If a permit match is found, the packet is forwarded to the static mapping statements where the global address is changed to the mapped inside address. Otherwise, if a deny match entry exists, or no match at all exists, the packet is dropped.

Be aware that the **conduit** command always needs to be paired with a **static** command. Otherwise, traffic will not be translated through the PIX.

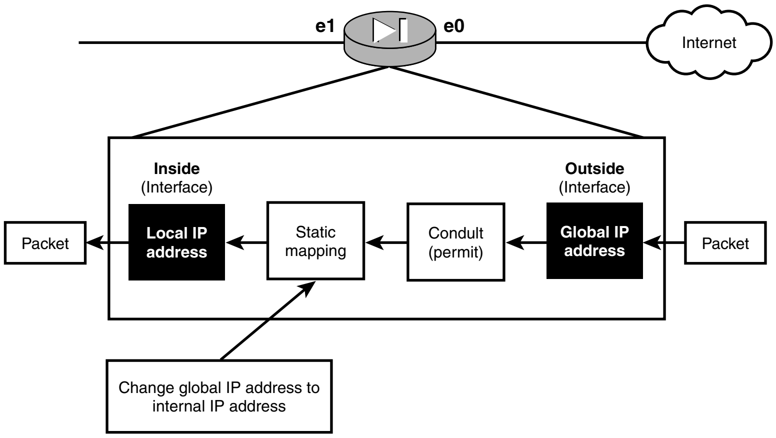

Figure 6.1 Traffic flow using a conduit.

As an example, if Peter, who is on the outside, wants to connect to Jack's computer on the inside using Telnet, the following steps must be taken:

1. Set up a `static` command to create a one-to-one mapping for Jack's computer to a global outside address.
2. Enter a `conduit` command to allow traffic from Peter's outside computer to connect to the global address that maps to Jack's inside computer.

Figure 6.2 shows Jack's and Peter's computers for the following command example.

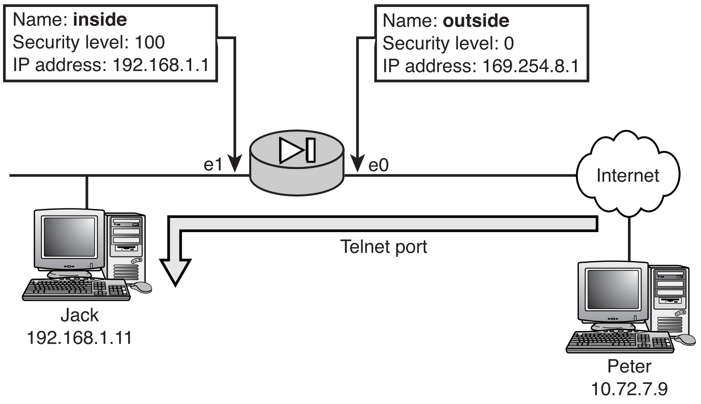

Figure 6.2 Outside traffic connecting to an inside computer.

The following Listing 6.1 allows Peter's computer to connect to Jack's computer:

Listing 6.1 static and conduit Commands

```
Pixfirewall(config)# static (inside, outside) 169.254.8.11 192.168.1.11
Pixfirewall(config)#
Pixfirewall(config)# conduit permit tcp
              ➥host 169.254.8.11 eq telnet host 10.72.7.9
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

Notice that the `conduit` command uses the global outside address instead of Jack's real internal address. All filters use the global address because, as traffic enters the PIX firewall, the filters take effect before the `static` command translates the global address to the real internal address.

The order of the source and destination parameters for the **conduit** command are in reverse compared to the standard **access-list** command. To illustrate, the commands are ordered as follows:

conduit:

```
conduit permit tcp (DESTINATION) (SOURCE)
```

access-list:

```
access-list 101 permit tcp (SOURCE) (DESTINATION)
```

The `conduit` command is very easy to use, but it does have some flaws. You might not have noticed that we didn't attach this command to the outside interface—or any interface, for that matter. This command is not defined against any single interface; it's applied to all interfaces with lower security levels. The conduit entries are also processed in the order in which you entered them into the system. So, if you entered a `permit conduit` command and then a `deny` command below it, the `deny` entry would never be processed; you would have to drop all the `conduit` entries and reenter them in the correct order.

A Basic conduit Command

Table 6.1 displays the field descriptions for the following `conduit` command syntax:

```
Pixfirewall(config)# [no] conduit permit ¦ deny protocol global_ip
              ➥global_mask [operator port [port]]
              ➥foreign_ip foreign_mask[operator port [port]]
```

Table 6.1 conduit Command Options

Option	Function
permit | deny	This defines whether to allow or disallow the matching traffic.
protocol	This defines the transport protocol, such as IP, TCP, UDP, or ICMP. (You use **ip** to specify all the protocols.)
global_ip global_mask	This is the global address you specified in the **static** command that eventually maps back to the internal address.
operator port	This is a comparison operand that lets you define single or multiple ports.
port	This enables you to specify the service you want to permit or deny. For example, you can use port 80 or the text **www** to allow HTTP traffic.
foreign_ip foreign_mask	This is the address of the outside computer trying to go through the PIX, sometimes known as the foreign address.

Access Control Lists

The access control list commands are used in just about every Cisco product to provide controlled access across a device or to define groups of addresses. We will talk here about access lists as named or numbered sets of entries that either permit or deny access across the PIX. A single access list can contain several, if not hundreds, of entries using `permit` or `deny` statements. These entries test traffic for source and destination address and ports matches. Access lists are created in the global area and can be attached to the desired interface needing control. Because they are created in the global area, a single access list can be attached (also known as *grouped*) to several interfaces simultaneously. If a change is made to the access list, all the interfaces using it are affected by that change. Figure 6.3 shows how ACLs filter the flow of traffic.

The ACL commands are similar to the commands on the Cisco IOS-based routers. They were introduced to the PIX firewall to provide more control, flexibility, and granularity and to support the standardization of Cisco commands across products. The ACL commands can perform the same functions as the `conduit` commands for traffic initiated on the outside needing to connect to inside computers. You can also use these commands to deny traffic you want to ensure is never allowed to enter a particular interface.

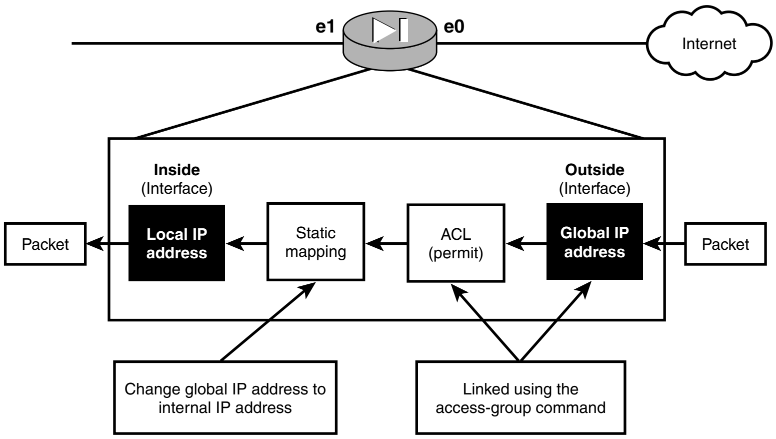

Figure 6.3 Traffic flow using access lists.

TIP

If you are using both the **access-list** command and the **conduit** command at the same time, the **access-list** command is processed first. Remember: Cisco has a large preference to the ACL commands, so use them instead of **conduit** commands when you can.

Access list commands also are slightly different from the normal IOS commands. PIX uses a normal mask and not a wildcard mask, as IOS routers do.

Configuring the `access-list` command to allow traffic inside involves three main steps:

1. Create a static mapping.
2. Set up the access list.
3. Attach it to the interface using the `access-group` command.

Working with the example in Figure 6.2, in which Peter wants to Telnet into Jack's computer, let's look at the syntax involved, only this time using an access list. The syntax to accomplish this task is similar to that of the `conduit` command, but the `access-list (SOURCE) (DESTINATION)` parameter ordering is different.

Listing 6.2 enables Peter to access Jack's computer using Telnet.

Listing 6.2 static and access-list Commands

```
Pixfirewall(config)# static (inside, outside) 169.254.8.11 192.168.1.11
Pixfirewall(config)# access-list Let-Peter-In permit tcp host 10.72.9.7
               ➥host 169.254.8.11 eq telnet
Pixfirewall(config)# access-group Let-Peter-In in interface outside
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

The `access-list` command in Listing 6.2 creates an access list called `Let-Peter-In`. The entry made in the list permits Telnet traffic coming from Peter (the source) to Jack's statically mapped global address (the destination). The `access-group` command attaches the list to the outside interface for all inbound traffic. The `access-group` command on the PIX can attach access lists only to the incoming traffic, not outgoing traffic as with IOS-based routers.

TIP

If you want to add more entries to an access list—for instance, to allow three computers access to the inside—just create the entries with the same name. The list will be processed in a top-down order. These commands create a list with three entries:

```
access-list 100 permit ip host 192.168.1.11 host 1.1.1.1
access-list 100 permit ip host 192.168.1.11 host 1.1.1.2
access-list 100 permit ip host 192.168.1.11 host 1.1.1.3
```

Access Control List Commands

Before we show you any more examples of access lists, a review of the details of the basic commands is necessary. Table 6.2 displays a list of several commands used to view, create, and delete access lists or access groups.

Table 6.2 ACL-related Commands

Command	Description
show access-list	Displays either all the access lists or a single access list
show access-group	Displays access groups
access-list	Used to create, append, or delete an access list
access-group	Used to attach an access list to or remove it from an interface
clear access-list	Used to delete either all the access lists or a single list
clear access-group	Used to delete either all the access groups or a single group

The show access-list Command

The `show` commands enable you to display access list entries and their respective hit counts. The hit count feature is quite useful. As traffic that matches an access list entry travels through an interface, this small counter increments and gives you an indication of how frequently your access list entry is being used. For example, the `show access-list` command in Listing 6.3 displays access list 100 and reveals that the first entry has been hit seven times, the second entry has been hit only four times, and the third has never been hit.

Listing 6.3 The show access-list Command

```
pixfirewall# show access-list 100
access-list 100; 3 elements
access-list 100 permit ip host 192.168.1.11 host 1.1.1.1 (hitcnt=7)
access-list 100 permit ip host 192.168.1.11 host 1.1.1.2 (hitcnt=4)
access-list 100 permit ip host 192.168.1.11 host 1.1.1.3 (hitcnt=0)
```

If you place a **deny IP any** any command as the last line of an access list, it will provide a hit count for all the traffic not matching the previous entries.

The **show access-group** Command

The `show access-group` command enables you to display all the access lists that are attached in an inward direction on all interfaces.

Listing 6.4 shows that access list 100 is attached to the inbound traffic on the outside interface and access list 1 is attached to the inside interface.

Listing 6.4 The show access-group Command

```
pixfirewall# show access-group
access-group 100 in interface outside
access-group 1 in interface inside
```

Access lists are bound to the inbound direction of an interface, not the outbound direction. If you want to control traffic leaving the outside interface, you must bind an ACL to the inbound traffic on the inside interface.

The **access-list** Command

The `access-list` command creates and deletes access lists and access list entries. An access list can have a single entry or several entries that are processed in the order in which you added them to the list; sometimes the hardest part about working with access lists is ensuring that the entries are in the order you need to provide the desired result. If you put the entries in the wrong order, you'll need to delete the entire list and start again. Note that you can delete single entries from an access list, so this feature can be a little helpful.

When access lists are created, they exist in a global area of the PIX firewall, meaning they can be attached to several interfaces simultaneously if you want. The command format for `access-list` commands is shown in Listing 6.5, and the `access-list` options are listed in Table 6.3.

Listing 6.5 access-list Command Syntax

```
Pixfirewall(config)# [no] access-list id permit¦deny protocol
                ➥source_ip_address
                ➥source_subnet_mask [operator port]
                ➥destination_ip_address
                ➥destination_subnet_mask [operator port]
```

Table 6.3 access-list Command Options

Option	Function
id	This is the number or name of the access list you are creating or appending to.
permit | deny	This defines whether to allow or disallow the matching traffic.
protocol	This defines the transport protocol, such as IP, TCP, UDP, or ICMP.
source_ip_address	This specifies the source IP address to detect. The keyword **any** can be used to specify all the source addresses, and the keyword **host** can be used to specify an exact address match.
operator port	This is a comparison operand that lets you define which ports will pass the acceptance criteria: **eq** is equals; **lt** is less than; and **gt** is greater than.
port	This allows you to specify the service you want to permit or deny. For example, you can use port 80 or the text **www** to allow HTTP traffic. Other keywords include **http**, **ident**, **nttp**, **ntp**, **pop2**, **pop3**, **rpc**, **smtp**, **snmp**, **snmptrap**, **sqlnet**, **telnet**, **tftp**, and **www**.
destination_ip_address	This is the destination address you are checking for. Be sure you are using the global address and not the real internal address—unless, of course, you are using the NAT 0 and bypassing translation.

In Listing 6.6, the access list named `101` is created with three entries.

Listing 6.6 Creating Three Entries in an Access List

```
Pixfirewall(config)# access-list 101 permit tcp host 10.10.12.37
                ➥host 169.254.8.1 eq telnet
Pixfirewall(config)# access-list 101 permit tcp host 10.10.12.27
                ➥host 169.254.8.1 eq www
Pixfirewall(config)# access-list 101 deny ip any any
```

Listing 6.6 creates an access list with three entries. The first entry states that traffic from source 10.10.12.37 to the destination (global) address of 169.254.8.1 port 23 (`telnet`) is permitted. The second command permits traffic from source 10.10.10.27 to the destination address of 169.254.8.1 port 80 (`www`). The last entry denies any source to any destination.

This example removes only a single entry from the access list named `101`:

```
Pixfirewall(config)# no access-list 101 deny ip any any
```

Using the `no` statement in front of the `access-list` command without specifying an individual entry allows you to remove the entire list from the system. The following command removes the access list named `101`:

```
Pixfirewall(config)# no access-list 101
```

The access-group Command

The `access-group` command enables you to attach and remove an access list from an interface. Access groups allow you to attach only a single ACL to the inbound direction of the interface. Attaching an ACL to the outbound direction is not an option on the PIX firewalls yet. The command format is as follows, and the command options are shown in Table 6.4:

```
Pixfirewall(config)# [no] access-group <access-list> in interface <if_name>
```

Table 6.4 access-group Command Options

Option	Function
access-list	This is the name or number of the access list you created, sometimes known as the ID.
in	Although this defines the direction of inbound traffic, *in* is in fact the only direction you can control.
interface	This is a required word that is used to specify to which interface you want the access list to be attached.
if_name	This defines the interface to which you want to attach the access list. For example, you can specify **outside**, **inside**, **dmz**, or whatever you've called your interface.

Here's how you use the `access-group` command:

```
Pixfirewall(config)# Access-group Let-Peter-In in interface outside
```

In the previous example, the access list named `Let-Peter-In` is bound to the inbound direction of the outside interface.

The following command deletes an access group binding:

```
Pixfirewall(config)# No Access-group Let-Peter-In in interface outside
```

The previous command removes the access list named `Let-Peter-In` from the outside interface. The access list itself is not deleted; only the link between the interface and the list is removed.

The clear access-list Command

The `clear access-list` can apply to either a single ACL or all the ACLs in your PIX. So be careful with this command! You might find yourself deleting all your ACLs and looking for that TFTP backup of your configuration you made last year (or hopefully more recently!). You can also use the `no access-list` command to delete a single access list. The `clear access-list` command's option is listed in Table 6.5, and its command format is as follows:

```
Pixfirewall(config)# Clear access-list [access-list]
```

Table 6.5 The clear access-list Command Option

Option	Function
[access-list]	This is the name or number of the access list you created, sometimes known as the ID.

Here's how you use the `clear access-list` and `no access-list` commands:

```
Pixfirewall(config)# Clear access-list Let-Peter-In
```

or

```
Pixfirewall(config)# no access-list Let-Peter-In
```

In the previous example, the `clear` command deletes the access list named `Let-Peter-In`. The alternative command to delete `Let-Peter-In` is the `no access-list` command.

The following demonstrates the `clear access-lists` command:

```
Pixfirewall(config)# clear access-list
```

This command removes all the access lists from the PIX firewall, so use this command with caution.

The clear access-group Command

The `clear access-group` command allows you to remove all access groups from your PIX firewall. If you want to delete only a single access group entry, you need to use the `no access-group` command. The command format is as follows:

```
Pixfirewall(config)# Clear access-group
```

An Access List Example

Now that you have learned about access lists, let's build a larger system using a three-pronged firewall such as the one in Figure 6.4. In this example, you will configure the entire firewall from the beginning to review the basic six commands. Then you will loosen up the firewall to allow traffic from the outside to access internal Web servers. Lastly, you'll allow Peter and Kristina to access Jack's computer via Telnet.

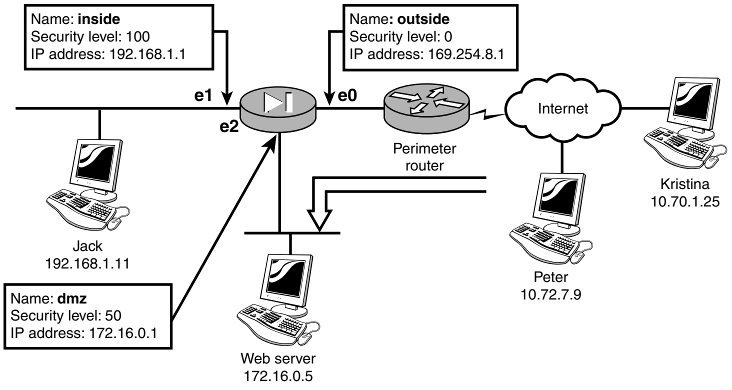

Figure 6.4 An access list with a three-pronged firewall.

Table 6.6 displays the basic commands needed to configure the firewall.

Table 6.6 The Six Basic Commands

Command	Description
nameif	Names the interface and sets the security levels.
interface	Defines the interface speed and duplex setting.
ip address	Sets the interface's IP address.
nat	Sets the NAT address ranges.
global	Sets the global range of addresses the NAT ID will use.
route	Sets the default route.

Listing 6.7 uses these six commands to configure the firewall shown previously in Figure 6.4.

Listing 6.7 Using the Six Basic Commands

```
Pixfirewall(config)#
Pixfirewall(config)# name-if e0 outside security0
Pixfirewall(config)# name-if e1 inside security100
Pixfirewall(config)# name-if e2 dmz security50
```

Listing 6.7 Using the Six Basic Commands *(continued)*

```
Pixfirewall(config)#
Pixfirewall(config)# interface e0 10baseT
Pixfirewall(config)# interface e1 10baseT
Pixfirewall(config)# interface e2 10baseT
Pixfirewall(config)#
Pixfirewall(config)# ip address outside 169.254.8.1 255.255.255.0
Pixfirewall(config)# ip address inside 192.168.1.1 255.255.255.0
Pixfirewall(config)# ip address dmz 172.16.0.1 255.255.0.0
Pixfirewall(config)#
Pixfirewall(config)# nat (inside) 1 192.168.1.0 255.255.255.0
Pixfirewall(config)# nat (dmz) 1 172.16.0.0 255.255.0.0
Pixfirewall(config)#
Pixfirewall(config)# global (outside) 1 169.254.8.10-169.254.8.253
              ➥255.255.255.0
Pixfirewall(config)#
Pixfirewall(config)# route outside 0.0.0.0 0.0.0.0 169.254.8.254 1
Pixfirewall(config)#
```

The sequence of commands in Listing 6.7 performs the following functions:

- The `name-if` command defines the interface name and security levels needed for the setup.
- The `interface` command sets all the interfaces to 10BASE-T.
- The `ip address` command sets the interface IP addresses.
- The first `nat` command allows the subnet 192.168.1.0 to be translated to an IP address in the global pool ID or 1.
- The second `nat` command allows the DMZ subnet 172.16.0.0 to be translated.
- The `global` command defines a range of global addresses that will be used by the `nat id 1` commands.
- The last command creates a default route to the perimeter router.

At this point, your firewall will allow traffic to pass in a single direction from higher security levels to lower security levels and the ASA will allow return traffic back through the PIX.

Now, let's allow traffic initiated from the outside to access the Web server located in the DMZ. We'll also need to allow Peter and Kristina to access Jack's computer using Telnet. The following are the three main steps you need to perform:

1. Create a static map of the global address 169.254.8.1 to map to the internal address of the Web server, 172.16.0.5. Then, you must map the global address of 169.254.8.2 to Jack's computer.

2. Next, you must create an access list that allows Web traffic from the outside to access the global address of 169.254.8.1. This address is mapped to the Web server. Then, you append to the access list Peter's and Kristina's addresses to allow Telnet access.
3. Finally, you use the access group to bind the ACL to the outside interface.

Listing 6.8 displays the code used in these three steps.

Listing 6.8 Configuring Traffic to Come In

```
Pixfirewall(config)# static (dmz, outside) 169.254.8.1 172.16.0.5
Pixfirewall(config)# static (inside, outside) 169.254.8.2 192.168.1.11
Pixfirewall(config)#
Pixfirewall(config)# access-list Let-Traffic-In permit tcp any
            ➥host 169.254.8.1 eq www
Pixfirewall(config)# access-list Let-Traffic-In permit tcp
            ➥host 10.70.1.25 host 169.254.8.2 eq telnet
Pixfirewall(config)# access-list Let-Traffic-In permit tcp
            ➥host 10.72.7.9 host 169.254.8.2 eq telnet
Pixfirewall(config)#
Pixfirewall(config)# access-group Let-Traffic-In in interface outside
Pixfirewall(config)#
Pixfirewall(config)#clear xlate
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# write memory
Pixfirewall(config)#
```

Listing 6.8 gives you the ability to allow traffic to the Web server and traffic to Jack's computer if it's from Peter or Kristina. The ACL must be bound to the outside interface, and a static address is used to translate the traffic coming in.

EXAM ALERT

Interfaces can have only one ACL attached to them in the inbound direction.

Controlling Traffic Going Out

We have been talking about controlling traffic coming into the firewall initiated from the outside. But the PIX can also control traffic heading toward the outside interface. To accomplish this, ACLs or outbound filter commands need to be placed on the higher security interfaces to permit or deny outbound traffic through that interface.

Using ACLs Going Out

Access control lists are the preferred method used by Cisco to control traffic flowing into the PIX. However, you cannot use ACLs on interfaces in the outbound direction as you can on IOS routers. Therefore, if you need to control traffic leaving the outside interface, you must attach the access list to the inside inbound interface, thus blocking the traffic before it gets to the outbound interface.

For example, if you wanted to prevent Jack's computer from reaching the Web site of 169.254.39.39, you would use the commands shown in Listing 6.9.

Listing 6.9 Blocking a Single Destination

```
Pixfirewall(config)# Access-list stop-jack deny IP host 192.168.1.11
              ➥host 169.254.39.39
Pixfirewall(config)# Access-list stop-jack permit IP any any
Pixfirewall(config)#
Pixfirewall(config)# Access-group stop-jack in interface inside
Pixfirewall(config)#
Pixfirewall(config)# Clear xlate
```

To prevent a whole subnet of 192.168.8.0 from accessing the PIX, the commands is Listing 6.10 could be used.

Listing 6.10 Blocking a Subnet

```
Pixfirewall(config)# Access-list stop-Sub deny IP 192.168.8.0
              ➥255.255.255.0 any
Pixfirewall(config)# Access-list stop-Sub permit IP any any
Pixfirewall(config)#
Pixfirewall(config)# Access-group stop-Sub in interface inside
Pixfirewall(config)#
Pixfirewall(config)# Clear xlate
```

Use access lists on the inside interface to prevent traffic from traveling across the PIX in the outbound direction.

Filtering Outbound Traffic

The `outbound` command is an older command that can be used to control traffic from higher security level interfaces to lower security level interfaces. The command is similar to the `conduit` command, but in the opposite direction. Also similar to the `conduit` command, it's being replaced by the `access-list` command. We will list the command only once here just to cover its basics. More information about this old command can be found at Cisco's Web site (`www.cisco.com`).

These two steps are required to set up an outbound command:

1. Create the outbound filter.
2. Attach the outbound filter to an interface.

The **outbound** Command

The command used to create these filters is explained in Table 6.7. Its syntax is as follows:

```
Pixfirewall(config)# [no] outbound <outbound_id> permit¦deny¦except
    IP_address [<mask> [port[-port]] [<protocol>]]
```

Table 6.7 outbound Command Options

Option	Function
outbound_id	Lists the number for the outbound filter
permit|deny|except	Permits, denies, or makes an exception
ip_address	The IP address of the internal address or the destination address that you want to block
port protocol	The port and the protocol

The **apply** Command

The second part of the outbound filter is the `apply` command, which attaches it to an interface. The following is the `apply` command's syntax:

```
Pixfirewall(config)# [no] apply [(<if_name>)] <outbound_id>
              ➥outgoing_src¦outgoing_dest
```

The Table 6.8 displays all the options for the `apply` command.

Table 6.8 apply Command Options

Option	Function
if_name	Name of the interface to which you want to attach the outbound ID
outbound_id	The ID number used to identify an outbound list
outgoing_src	States that the addresses in the **outbound** command are the internal source addresses that will be denied or permitted
outgoing_dest	States that the addresses in the **outbound** command are destination addresses that will be denied or permitted

Outbound Filter Example

In Listing 6.11, address 192.168.1.11 and address 192.168.1.12 are allowed to pass but all other outbound traffic is denied.

Listing 6.11 apply and outbound Example

```
Pixfirewall(config)# outbound 1 deny 0.0.0.0 0.0.0.0 0 0
Pixfirewall(config)# outbound 1 permit 192.168.1.11 255.255.255.255 0 0
Pixfirewall(config)# outbound 1 permit 192.168.1.12 255.255.255.255 0 0
apply (inside) 1 outgoing_src_
```

Note the `outbound` command doesn't follow the order in which you entered the commands as the ACL does; it actually reorders the entries. This makes it very difficult to get used to and takes proper planning before you use it.

Turbo ACLs

Software release 6.2 introduced a new feature called *turbo ACLs*. Turbo ACLs decrease the time it takes to scan through large access lists. Large access lists take longer to process because every entry might need to be scanned for a possible match. The longer it takes to scan the access list, the slower your traffic will be.

Turbo ACLs create a compiled index against large access lists that contain 19 or more entries. This index is similar to a database index or an index in a book. The PIX scans the index for a match instead of the list itself. This reduces the time it takes to search for possible matches in an ACL, making your throughput faster.

Some of the requirements needed for turbo ACLs are as follows:

- Software release 6.2 and above
- Minimum of 2.1MB of available flash
- A PIX firewall that has 16MB or more flash memory
- Access lists with 19 or more entries

Turbo ACLs speed things up but are very memory intensive, requiring 2.1MB of free memory. Therefore, smaller PIX firewalls such as the 501 cannot use turbo ACLs because they don't have enough free memory in flash.

EXAM ALERT

Turbo ACLs are simple to create and work on all models of the PIX except the 501. The 501 does not support turbo ACLs—turbo ACLs are typically not used on smaller firewall models because they use too much memory.

The turbo ACLs command is `access-list compile`, and the following is an example of compiling all your access lists:

```
Pixfirewall(config)# access-list compiled
```

You can also be selective on which access lists you compile by placing the name of the ACL in the command, as shown here:

```
Pixfirewall(config)# access-list Let-Peter-In compiled
```

To view turbo ACLs, you use the `show access-list` command; to delete a compiled access list, you use the `no access-list compiled`.

Turbo ACLs work best on access lists with 19 or more entries. If a list has fewer than 19 entries, you really don't get any speed increase by reading the index instead of the list itself.

The Basics of Object Grouping

Another new feature introduced in version 6.2 is object grouping for access control lists and `conduit` commands. *Object grouping* creates groups of networks, services, ICMP, and protocols that allow themselves to be joined together in access lists, conduits, or members of other object groups.

Object groups enable you to save the amount of access list entries needed to create large ACLs by allowing you to reference object groups. For example, a typical access list that needs 5 IP addresses and 3 services (such as Telnet, WWW, and FTP) for each would require 15 ACL entries. If you used object grouping, you would need only 1 object group for the 5 IP addresses (`network`) and 1 object group for the 3 ports (Telnet, WWW, and FTP [`services`]). Then, you could join these 2 groups with a single access list. The result of the 2 groups would produce a large list of 15 entries that contains every combination of IP address and services.

When working on large, complex access lists, object groups enable you to save on the number of entries needed to create the access list.

Figure 6.5 displays an access list joining two object groups into one access list entry that contains every combination of the two object groups.

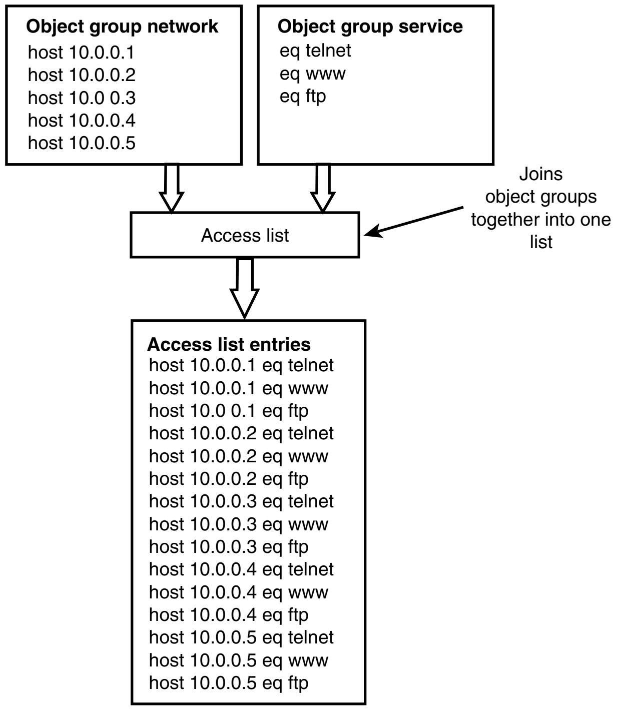

Figure 6.5 Object groups joined with an access lists.

Types of Object Groups

The four types of object group commands that can be used are `network`, `service`, `protocols`, and `icmp-types`. Table 6.9 displays the object group commands.

Table 6.9 Types of object-group Commands

Command	Description
object-group network	Defines a group of hosts or subnets
object-group services	Defines a group of TCP and UDP port numbers
object-group protocol	Defines a group of IP protocols, such as IP, ICMP, TCP, and UDP
object-group icmp-type	Defines a group of ICMP messages

The object group commands listed in Table 6.9 place you into a subconfiguration mode. To leave this mode, just type **`exit`** to return to the normal configuration mode prompt.

Know the four types of object groups: **network**, **service**, **protocol**, and **icmp-type**. Also, be sure you know the commands needed to create them.

Object Group Networks

Network groups are used to create large lists of hosts or networks that can be used in access list commands. The command sequence is shown in Listing 6.12, and the options are listed in Table 6.10.

Listing 6.12 object-group Network Commands

```
Pixfirewall(config)# object-group network obj_grp_id
Pixfirewall(config-network)# description
Pixfirewall(config-network)# network-object host host_address
Pixfirewall(config-network)# network-object network_address subnet_mask
Pixfirewall(config-network)# group-object
```

Table 6.10 object-group network Command Options

Option	Function
obj_grp_id	Defines the name of the object group you are creating or editing
description	Sets a description to the object group
network-object	Specifies the host or network you are listing
group-object	Allows you to reference another network object group

Listing 6.13 displays a network object group called `TheNetworkList` being created. The description is set, and four host addresses and one subnet entry are added to the group.

Listing 6.13 Example of the object-group network Command

```
pixfirewall(config)# object-group network TheNetworkList
pixfirewall(config-network)# description This is my great network list
pixfirewall(config-network)# network-object host 10.0.0.1
pixfirewall(config-network)# network-object host 10.0.0.2
pixfirewall(config-network)# network-object host 10.0.0.3
pixfirewall(config-network)# network-object host 10.0.0.4
pixfirewall(config-network)# network-object 11.0.0.0 255.0.0.0
pixfirewall(config-network)# exit
pixfirewall(config)# show object-group id TheNetworkList
object-group network TheNetworkList
  description: This is my great network list
  network-object host 10.0.0.1
  network-object host 10.0.0.2
  network-object host 10.0.0.3
  network-object host 10.0.0.4
  network-object 11.0.0.0 255.0.0.0
pixfirewall(config)#
```

Object Group Services

Service groups are used to create lists of TCP and UDP port number services, such as Telnet, WWW, and FTP. The command sequence is shown in Listing 6.14.

Listing 6.14 object-group Services Commands

```
Pixfirewall(config)# object-group service obj_grp_id tcp¦udp¦tcp-udp
Pixfirewall(config-service)# description
Pixfirewall(config-service)# port-object eq¦range
Pixfirewall(config-service)# group-object
```

Table 6.11 displays the commands and syntax needed to support the `object-group service` command.

Table 6.11 object-group services Command Options

Option	Function
obj_grp_id	Defines the name of the object group you are creating or editing
tcp|udp|tcp-udp	Defines the service protocol to create
description	Sets a description to the object group
port-object	Specifies the exact port using the **eq** operator or a range of port numbers with the **range** operator
group-object	Allows you to reference another service object group

Listing 6.15 displays a service object group called `ThePortList` being created. This group sets the description and creates matching entries for Telnet, WWW, and FTP ports. Lastly, a range of ports from 1433 to 1435 is set.

Listing 6.15 Example of the object-group service Command

```
pixfirewall(config)# object-group service ThePortList tcp
pixfirewall(config-service)# description This is my great port list
pixfirewall(config-service)# port-object eq telnet
pixfirewall(config-service)# port-object eq www
pixfirewall(config-service)# port-object eq ftp
pixfirewall(config-service)# port-object range 1433 1435
pixfirewall(config-service)# exit
pixfirewall(config)# show object-group id ThePortList
object-group service ThePortList tcp
  description: This is my great port list
  port-object eq telnet
  port-object eq www
  port-object eq ftp
  port-object range 1433 1435
pixfirewall(config)#
```

Object Group Protocols

Protocol groups enable you to create a group of protocols such as IP, TCP, UDP, or ICMP. This object group can be used in the protocol portion of an access list command. The command sequence is shown in Listing 6.16.

Listing 6.16 object-group protocol Commands

```
Pixfirewall(config)# object-group protocol obj_grp_id
Pixfirewall(config-protocol)# description
Pixfirewall(config-protocol)# protocol-object protocol
Pixfirewall(config-protocol)# group-object
```

Table 6.12 displays the command options for the `object-group protocol` command.

Table 6.12 object-group protocol Command Options

Option	Function
obj_grp_id	Defines the name of the object group you are creating or editing.
description	Sets a description to the object group.
protocol-object	Specifies the protocol either by name or number. Sample protocols are IP, TCP, UDP, and ICMP.
group-object	Allows you to reference another protocol object group.

Listing 6.17 displays a protocol object group called `TheProtocolList` being created. This group sets the description and creates three entries for TCP, UDP, and GRE protocols.

Listing 6.17 Example of the object-group protocol Command

```
pixfirewall(config)# object-group protocol TheProtocolList
pixfirewall(config-protocol)# description This is my great protocol list
pixfirewall(config-protocol)# protocol-object tcp
pixfirewall(config-protocol)# protocol-object udp
pixfirewall(config-protocol)# protocol-object gre
pixfirewall(config-protocol)# exit
pixfirewall(config)# show object-group id TheProtocolList
object-group protocol TheProtocolList
  description: This is my great protocol list
  protocol-object tcp
  protocol-object udp
  protocol-object gre
pixfirewall(config)#
```

EXAM ALERT

After the **object-group protocol FastStuff** command, the next line takes you into the configuration for that object group (**FastStuff**). The command prompt displays **pixfirewall(config-protocol)#**.

ICMP Groups

ICMP groups enable you to create groups based on ICMP messages. Listing 6.18 shows the syntax for this command.

Listing 6.18 object-group icmp Commands

```
Pixfirewall(config)# object-group icmp-type obj_grp_id
Pixfirewall(config-icmp-type)# description
Pixfirewall(config-icmp-type)# icmp-object type
Pixfirewall(config-icmp-type)# group-object
```

Table 6.13 displays the options for the `object-group icmp` command.

Table 6.13 object-group icmp Command Options

Option	Function
obj_grp_id	Defines the name of the object group you are creating or editing
description	Sets a description to the object group
icmp-object	Specifies the type of ICMP message—for example, **echo** and **echo-reply**
group-object	Allows you to reference another ICMP object group

Listing 6.19 displays an ICMP object group called `TheICMPList` being created. This group sets the description and creates two ICMP entries: `echo` and `echo-reply`.

Listing 6.19 Example of the object-group icmp-type Command

```
pixfirewall(config)# object-group icmp-type TheICMPList
pixfirewall(config-icmp-type)# description This is my great icmp list
pixfirewall(config-icmp-type)# icmp-object echo
pixfirewall(config-icmp-type)# icmp-object echo-reply
pixfirewall(config-icmp-type)# exit
pixfirewall(config)# show object-group id TheICMPList
object-group icmp-type TheICMPList
  description: This is my great icmp list
  icmp-object echo
  icmp-object echo-reply
pixfirewall(config)#
```

Displaying Object Groups

To display existing object groups, you can use the `show object-group` commands, which are explained in Table 6.14.

Table 6.14 show object-group Commands

Command	Description
show object-group	Displays all the object groups
show object-group [protocol | service | icmp-type | network]	Displays the object group based on the type, such as network, service, protocol, or ICMP type
show object-group id <obj_grp_id>	Displays the object group ID with the matching name

Deleting Object Groups

To delete an object group, you must first remove it from any other reference that might be using it. For example, if the object group `TheICMPList` is being used by an access list, you must remove it from the access list before you can delete it. Table 6.15 displays the commands used to delete object groups.

Table 6.15 clear object-group Commands

Command	Description
clear object-group	Deletes all the object groups
clear object-group [protocol | service | icmp-type | network]	Deletes all the object groups of a specific type, such as network, service, protocol, or ICMP type
no object-group [protocol | service | icmp-type | network] name	Removes a single object group by name and type

The command shown here deletes the object group called `TheICMPList`:

```
pixfirewall(config)#  no object-group icmp-type TheICMPList
```

Nesting Object Groups

Object groups can also be nested inside other object groups. In Figure 6.6, two network groups are referenced in a third network object group. This new group could then be used by an access list or `conduit` command to produce the result of all the addresses.

One restriction is that you can only nest groups of the same type. For example, network groups can be nested only into other network groups.

To demonstrate, Listing 6.20 shows how two groups can be nested into a third group.

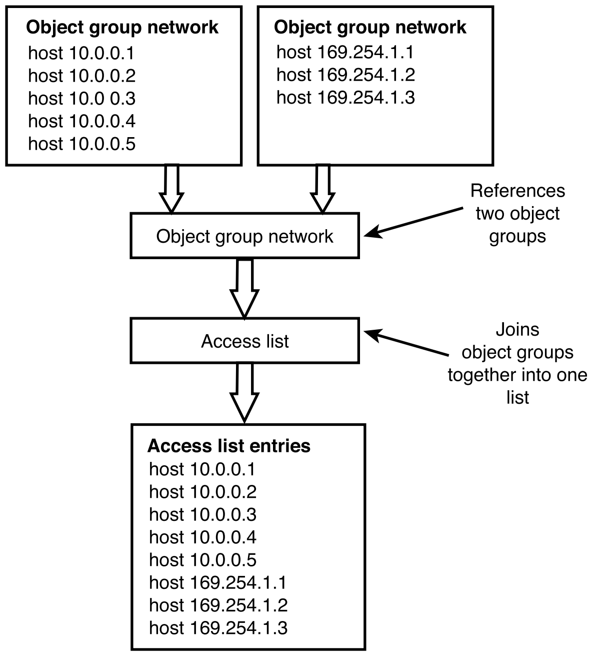

Figure 6.6 Object groups nested within an object group.

Listing 6.20 Nesting object-group Commands Example

```
pixfirewall(config)# object-group network TheIPList1
pixfirewall(config-network)# description This is my great list 1
pixfirewall(config-network)# network-object host 10.0.0.1
pixfirewall(config-network)# network-object host 10.0.0.2
pixfirewall(config-network)# network-object host 10.0.0.3
pixfirewall(config-network)# network-object host 10.0.0.4
pixfirewall(config-network)# network-object host 10.0.0.5
pixfirewall(config-network)# exit
pixfirewall(config)#
pixfirewall(config)# object-group network TheIPList2
pixfirewall(config-network)# description This is my great list 2
pixfirewall(config-network)# network-object host 169.254.1.1
pixfirewall(config-network)# network-object host 169.254.1.2
pixfirewall(config-network)# network-object host 169.254.1.3
pixfirewall(config-network)# exit
pixfirewall(config)#
pixfirewall(config)# object-group network TheIPList3
pixfirewall(config-network)# description This is my great list 3
pixfirewall(config-network)# group-object TheIPList1
pixfirewall(config-network)# group-object TheIPList2
pixfirewall(config-network)# exit
pixfirewall(config)#
```

Listing 6.20 shows that the object group `TheIPList3` contains references to both the `TheIPList1` and `TheIPList2` object groups.

ICMP Through the PIX Firewall

By default, the PIX firewall blocks all ICMP traffic such as trace routes and pings. The two points at which ICMP traffic is directed are when passing through the PIX and when directed at the PIX. The PIX blocks both. Table 6.16 displays some of the ICMP message numbers that can be used on the PIX.

Table 6.16 ICMP Message Numbers

Message Number	Cisco Parameter
0	**echo-reply**
3	**unreachable**
4	**source-quench**
5	**redirect**
8	**echo**
11	**time-exceeded**
12	**parameter-problem**
13	**timestamp-request**
14	**timestamp-reply**
15	**information-request**
16	**information-reply**

Outbound ICMP Traffic

The PIX allows ICMP traffic to pass from higher security levels to lower security levels but blocks ICMP traffic from lower security level interfaces to higher security level interfaces. To allow ICMP traffic to flow both ways, you need to configure a static translation and use a `conduit` or ACL command.

The following two steps are needed to permit returning ICMP ping traffic to pass back to the source:

1. Create a static mapping.
2. Use an `access-list` or `conduit` command to allow ICMP traffic to pass. If you use the `access-list` command, you also must use the `access-group` command to link it to an interface.

Listing 6.21 demonstrates how to permit return traffic from the outside interface to ping traffic on the inside.

Listing 6.21 Using the conduit Command to Allow ICMP Traffic In

```
Pixfirewall(config)# static (inside,outside) 169.254.8.1 192.168.1.11
              ➥netmask 255.255.255.255 0 0
Pixfirewall(config)#
Pixfirewall(config)# conduit permit icmp 169.254.8.1 255.255.255.255
              ➥0.0.0.0 0.0.0.0 echo-reply
Pixfirewall(config)# conduit permit icmp 169.254.8.1 255.255.255.255
              ➥0.0.0.0 0.0.0.0 source-quench
Pixfirewall(config)# conduit permit icmp 169.254.8.1 255.255.255.255
              ➥0.0.0.0 0.0.0.0 unreachable
Pixfirewall(config)# conduit permit icmp 169.254.8.1 255.255.255.255
              ➥0.0.0.0 0.0.0.0 time-exceeded
```

Listing 6.22 demonstrates how to permit return traffic from the outside interface to ping traffic on the inside using the `access-list` command.

Listing 6.22 Using the access-list Command to Allow ICMP Traffic In

```
Pixfirewall(config)# static (inside,outside) 169.254.8.1 192.168.1.11
              ➥netmask 255.255.255.255 0 0
Pixfirewall(config)#
Pixfirewall(config)# access-list 100 permit icmp any host 169.254.8.1
              ➥echo-reply
Pixfirewall(config)# access-list 100 permit icmp any host 169.254.8.1
              ➥source-quench
Pixfirewall(config)# access-list 100 permit icmp any host 169.254.8.1
              ➥unreachable
Pixfirewall(config)# access-list 100 permit icmp any host 169.254.8.1
              ➥time-exceeded
Pixfirewall(config)# access-group 100 in interface outside
```

ICMP Directed at the PIX

The PIX firewall can be set to block ICMP traffic directed at it. This allows the PIX to reject responses to ICMP requests; as a result it is more difficult for hackers to discover the firewall. To allow the firewall to respond to ICMP traffic requests, you must use the `icmp` command instead of an ACL command. The following is the syntax for the `icmp` command:

```
pixfirewall(config)# [no] icmp permit¦deny <ip-address> <net-mask>
              ➥ [<icmp-type>] <if-name>
pixfirewall(config)# [clear¦show] icmp
```

Table 6.17 displays the options available for the `icmp` command.

Table 6.17 icmp Command Options

Option	Function
permit|deny	This permits or denies ICMP traffic.
ip-address	This is the source IP address of the ICMP traffic.
net-mask	This is the mask of the allowed traffic. 255.255.255.255 would allow only a specific IP address.
icmp-type	This is the type of ICMP traffic, such as **echo-reply** or **unreachable**.
if-name	This is the name of the interface to which the ICMP entry is applied.

The following command allows the outside interface to respond to ping requests on the outside interface:

```
icmp permit any echo outside
```

To deny ICMP traffic on the outside interface, the following command is necessary:

```
icmp deny any echo outside
```