5

Translations and Connections

Terms you'll need to understand:

- ✓ Translation table
- ✓ Connection table
- ✓ TCP
- ✓ Embryonic connection
- ✓ UDP
- ✓ Static mapping
- ✓ Dynamic mapping
- ✓ Port redirection

Techniques you'll need to master:

- ✓ The **clear xlate** command
- ✓ The **show xlate** command
- ✓ The **show conn** command
- ✓ The **static** command
- ✓ NAT and PAT
- ✓ **nat 0**
- ✓ Port redirection

The PIX firewall allows traffic to flow from higher security levels to lower security levels using features such as network address translation (NAT), port address translation (PAT), and static mappings. Traffic originating from lower security level interfaces destined to higher security level interfaces must be manually configured using the `static` and `conduit` commands before the traffic can pass. This chapter covers how to use the PIX firewall features to allow traffic to flow between interfaces.

Transport Protocols (Layer 4)

When programmers create applications that need to communicate across a network, they typically choose the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), or both protocols. These protocols help applications deliver data in either a reliable, connection-oriented (TCP) fashion or an unreliable, fast, connectionless-oriented (UDP) fashion. The PIX firewall tracks the TCP or UDP traffic traversing the firewall in slightly different ways, but both are monitored to provide users with stateful sessions across the firewall.

TCP Overview

TCP is a connection-oriented protocol. Connection-oriented protocols provide reliability that guarantees the delivery of data to its destination and that has enough know-how to retransmit missing data. TCP contains a defined state machine mechanism similar to a modem calling another computer. The following are the three states of TCP:

1. Set up the connection.
2. Send the data.
3. Disconnect the connection.

During the connection setup state, a process known as the *three-way handshake* takes place. This handshake helps the two communicating computers establish connection parameters that will be used during the transmission of data. The first step of the handshake opens a connection by sending a synchronization (SYN) request. This is sent to the destination computer, which in step two sends its own SYN request back with an acknowledgement (ACK) for the SYN it has received. In step three, the original computer responds to the SYN and ACK received with the last and final ACK. At this point the two computers are ready to send data. After the data transfer is finished, a FIN message is sent to disconnect the session.

For example, in Figure 5.1 Jack wants to communicate with Peter's computer. The following steps take place:

1. Jack sends a synchronization message to open a connection.
2. Peter's computer responds with its own SYN to Jack and an acknowledgement to Jack's SYN.
3. Jack receives Peter's SYN and ACK and responds with a final ACK back to Peter.
4. Data transmission begins.
5. The computers are disconnected by the FIN message being sent.

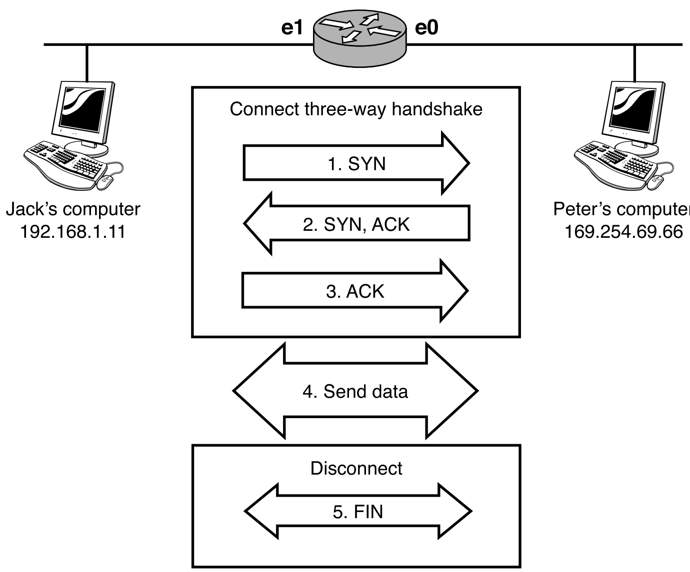

Figure 5.1 A three-way handshake.

How the PIX Works with TCP

The PIX is a stateful firewall that monitors the traffic flowing through it. The TCP protocol is very predictable, and the PIX system uses this to its advantage. Using the previous example, when Jack first communicates with Peter, the PIX records the connection information into its connection table. It then allows the SYN and ACK messages from Peter to pass back through the firewall to Jack—and only Jack. After Jack sends the final ACK, the TCP session is established and a connection slot is established in the PIX that allows Jack and Peter to send data back and forth. If you were to view the connection table, you would see that a connection is being made between Jack and Peter. As they disconnect from each other, a FIN message is sent that closes each TCP session. The PIX monitors the traffic for this FIN message and uses it as a flag to remove the connection table entry created for the two. If Peter were to send more information to Jack after the FIN message has been sent, no entry would be found in the table and Peter's data would be discarded, thus protecting Jack from uninitiated traffic.

By default, the PIX firewall drops all initiated traffic from lower-level interfaces (Peter) to higher-level interfaces (Jack). A manual static entry can be added to override this, though.

Embryonic Connections

The process TCP uses is very predictable; hackers can use this to their advantage when trying to hack a system. For example a *SYN attack* occurs when a hacker sends hundreds or even thousands of SYN open connection requests to a computer, with the intention of never acknowledging those requests with an ACK. This leaves the victim computer wasting resources for each of those half-open connections. If too much of the victim's resources are wasted, the computer can fail, causing a denial of service. These half-open connections can be limited by using *embryonic connection values* on several of the PIX commands. Limiting the number of half-open connections helps guarantee that the computer won't be overloaded with false open requests.

UDP Overview

The User Datagram Protocol is a connectionless protocol. Connectionless protocols do not guarantee that data will be delivered to the destination through the use of SYN, ACK, or FIN messages, thus they're faster than TCP. If reliability is necessary, however, the application layer needs to monitor for missing data. UDP causes the PIX several problems. Because UDP does not use a three-way handshake or FIN messages, the PIX never really knows when to remove the connection slot entry between two computers. As a result, the PIX uses an idle timer to monitor whether traffic is passing between the two computers. If the timer expires before any traffic has passed, the connection is assumed ended and the connection slot entry is removed.

Private Addresses

Private address are IP addresses that are available for everyone to use inside a private company, home, or office. RFC 1918, "Address Allocation for

Private Internets," designates IP address ranges that have been blocked by IANA from use on the Internet.

Although you might think that private addresses are nonroutable, this is technically not true. These addresses work just fine in routing tables; they just do not exist—or should not exist—in routing tables on the Internet and thus cannot be routed anywhere. In the field, you will see these ranges used repeatedly because companies typically don't like to use real public Internet addresses inside their building or private networks. Instead, they prefer to use NAT to translate these nonroutable, private addresses to real public addresses that can be routed on the Internet. The list shown here displays the private IP address ranges that can be used:

- *Class A*—10.0.0.0–10.255.255.255 mask 255.0.0.0
- *Class B*—172.16.0.0–172.31.255.255 mask 255.240.0.0
- *Class C*—192.168.0.0–192.168.255.255 mask 255.255.0.0

Address Translation

The PIX firewall creates IP address-to-IP address mappings of traffic flowing through the firewall. Typically, the inside interface uses private addressing (RFC 1918) that cannot be used on the Internet. So, as the traffic is funneled out of the outside interface, NAT or PAT translation occurs. NAT replaces the source's private address with an available global public address on the outside interface, allowing traffic to travel on the public network while hiding the real (private) user IP address.

Translation (xlate) and Connection (conn) Tables

The PIX uses two main tables to track the traffic flowing through the system: the translation (xlate) table and the connection (conn) table. The translation table is used for IP address-to-IP address mapping and is commonly known as the xlate table. For example, as data from Jack travels across the firewall, his source address of 192.168.1.11 is changed to 169.254.8.1. These entries are sometimes called *slots*.

xlate table entries remain in the table for 180 minutes by default. You can use the **timeout xlate** command to change this setting.

The connection table contains layer 4 TCP or UDP sessions and is used to track with whom Jack has a current session. For example, as Jack sends data to Peter, a connection is made in the xlate table that represents the session generated between the two. After the two have finished sending data, the connection entry is automatically removed. Listing 5.1 shows the xlate and `conn` commands.

Listing 5.1 The show xlate and show conn Commands

```
Pixfirewall# show xlate
1 in use, 53 most used
PAT Global 169.254.8.1(1237) Local 192.168.1.11(1969)
Pixfirewall#
Pixfirewall# show conn
1 in use, 5 most used
TCP out 169.254.69.66:80 in 192.168.1.11:1969 idle 0:00:03 Bytes 334
Pixfirewall#
```

Listing 5.1 shows that the local address 192.168.1.11 (Jack) is being translated to 169.254.8.1. The `show conn` command shows that a TCP session exists between 192.168.1.11 (Jack) and IP address 169.254.69.66.

Step-by-Step Flow

The following demonstrates the step-by-step flow that occurs as Jack's computer connects to Peter's computer through the PIX firewall:

1. Jack wants to connect to Peter's computer, and the xlate slot is made for Jack to travel through the PIX. His IP address of 192.168.1.11 is translated to 169.254.8.1.
2. Jack opens a connection (SYN) with Peter (IP address 169.254.69.66).
3. The session between Jack and Peter is recorded in the conn table and the data is ready to be sent.
4. Jack and Peter send data back and forth.
5. The conn table entry is removed when either the TCP FIN message is seen or the conn timer expires.

Note that the xlate slot is not removed yet; if Jack wants to communicate with Peter or another computer, the xlate slot for 192.168.1.11 to 169.254.8.1 is reused. In Figure 5.2 Jack's computer is connecting to four Web servers on the Internet and the PIX is using NAT to translate addresses. The xlate table shows a single entry for the translation, and the conn table displays the four TCP sessions created to each Web site. Figure 5.2 displays Jack's xlate and connection information.

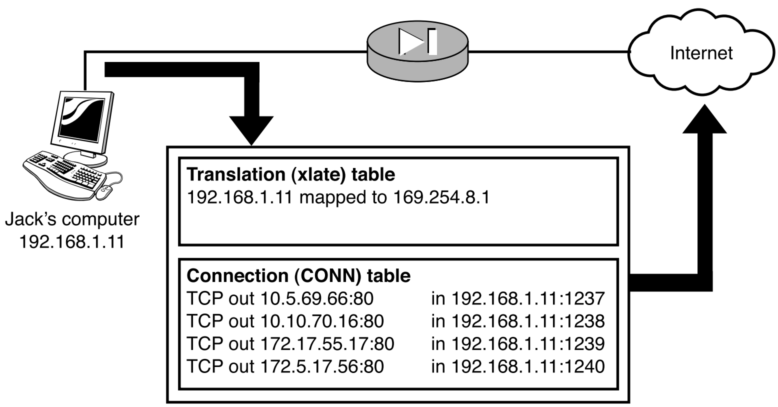

Figure 5.2 xlate and conn tables.

Whenever you make changes using the **access-lists**, **alias**, **global**, **interface**, **ip address**, **nameif**, **nat**, **outbound**, or **static** command, you should clear the xlate table using the **clear xlate** command.

The show and clear xlate Table Commands

The `xlate` command has several parameters that can be used with it. When using the `xlate` by itself, you display or clear the entire table. The parameters give you the granularity to work with only a few entries or groups of entries. The following is the `xlate` command's syntax:

```
show¦clear xlate [global¦local <ip1[-ip2]> [netmask <mask>]]
                [gport ¦lport <port1[-port2]>]
                [interface <if1[,if2]>]
                [state <static[,portmap][,norandomseq][,identity]>]
                [debug]
                [count]
```

Table 5.1 displays several `xlate` commands and their descriptions.

Table 5.1 xlate Commands

Command	Description
show xlate	Displays the contents of the xlate table
show xlate detail	Displays more detail about the entries in the table
show xlate state static	Displays only the static entries in the xlate table
show xlate count	Displays the current entries being used and the most frequently used ones
clear xlate	Clears the entire table
clear xlate state static	Clears only the static mappings in the table

The show conn Table Commands

The `conn` command is very similar to the `xlate` command; you can view or affect the entire table or just a subset of its entries. Remember that the connection table is used to monitor and control the sessions between two computers. Therefore, the parameters for the `conn` command include things such as FIN, TCP, and UDP. The command syntax is as follows:

```
show conn [count] ¦ [protocol <tcp¦udp>]
        [foreign¦local <ip1[-ip2]> [netmask <mask>]]
        [lport¦fport <port1[-port2>]
        [state <up[,finin][,finout][,http_get][,smtp_data]
                [,data_in][,data_out][,...]>]
```

Table 5.2 conn Commands

Command	Description
show conn	Displays the entire contents of the connection table
show conn detail	Displays detailed information about each connection entry
show conn protocol udp	Displays only the UDP traffic connection table entries

Outbound Traffic

The PIX firewall, by default, allows traffic from higher security level interfaces to traverse to interfaces with lower security level interfaces. This means that traffic initiated on the inside interface is allowed to pass to the outside interface. Setting up outbound traffic is a fairly easy task. You can use the `nat`, `global`, and `static` commands to help control which IP address translations you want to occur. You can also use commands such as `access-list` to control who can and cannot traverse your firewall. Also, you can use various methods, such as static and dynamic translation, to translate inside IP addresses to outside IP addresses.

To set up a basic scenario, use Figure 5.3 as the network diagram. Subnets 192.168.1.0/24 are the inside addresses that need to be translated to outside addresses.

TIP: No traffic is allowed to pass through the PIX until a **nat-global** or **static** command has been issued.

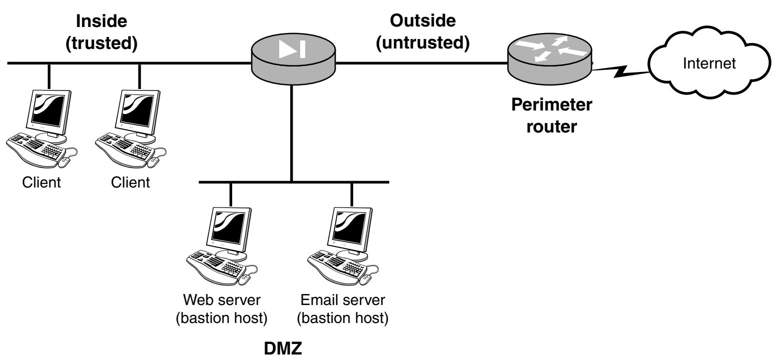

Figure 5.3 Network diagram.

Static Address Translations

To configure static NAT, you can use the `static` command, as shown here:

```
Pixfirewall(config)# [no] static [(internal_if_name, external_if_name)]
                {<global_ip>¦interface} <local_ip> [dns] [netmask <mask>]
                [<max_conns> [<emb_limit> [<norandomseq>]]]
```

ALERT: Although the **static** command lists the interface names in internal-external sequence, the sequence of the IP addresses that follow is reversed! Figure 5.4 shows how the internal inside interface address of 192.168.1.11 is mapped to the outside address of 169.254.8.1.

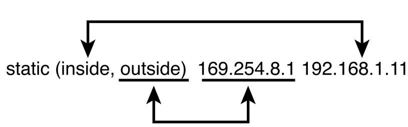

Figure 5.4 The **static** command syntax.

Table 5.3 static Command Options

Option	Function
internal_if_name	This defines the internal interface name that has a higher security level.
external_if_name	This defines the external interface name that has a lower security level.
global_IP	This is the global IP address you want your internal user to always use.
local_IP	This is the internal IP address you want statically assigned.
network_mask	The mask is used for both the internal and the external IP addresses. By using 255.255.255.255, you can define a specific host; by using 255.255.255.0, you can define an entire subnet.
max_conns	This defines the maximum number of connections an IP address can use at the same time.
em_limit	This defines the embryonic connection limit. The default is **0**, which means unlimited connections are allowed.
norandomseq	This allows you to turn off randomizing the TCP/IP packet sequence numbers. This is not recommended, but the parameter is available if necessary.

The `static` command enables you to bind an internal inside IP address to an outside global address, making a one-to-one mapping. Every time a specific inside user travels across the PIX, that user is assigned the same global address. For example, if Jack's internal IP address is 192.168.1.11, you can use the `static` command shown here to bind Jack's address to a specific global address, such as 169.254.8.1. The following command creates this one-to-one mapping:

```
Pixfirewall(config)# static (inside, outside) 169.254.8.1 192.168.1.11
```

Now, when Jack is traveling across the firewall, he will always use 169.254.8.1 on the public side and the PIX xlate table will show the following:

```
Pixfirewall# show xlate
1 in use, 1237 most used
Global 169.254.8.1 Local 192.168.1.11
```

Listing 5.2 creates a static one-to-one mapping from the three internal addresses to three global addresses. Figure 5.5 display the one-to-one mapping graphically.

Listing 5.2 Static One-to-One Mapping

```
Pixfirewall(config)# static (inside, outside) 169.254.8.1 192.168.1.11
Pixfirewall(config)# static (inside, outside) 169.254.8.2 192.168.1.12
Pixfirewall(config)# static (inside, outside) 169.254.8.3 192.168.1.13
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

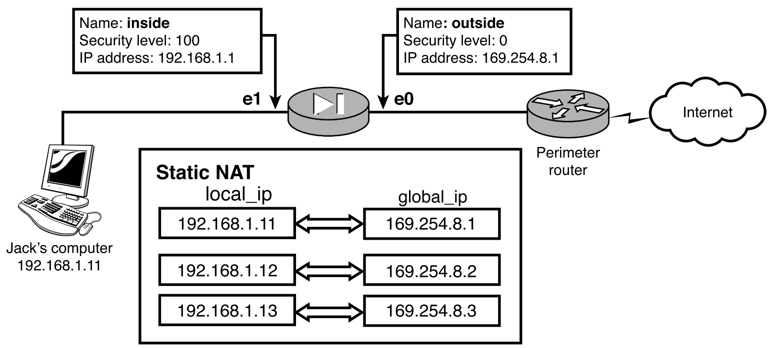

Figure 5.5 The **static** command.

Dynamic Address Translations

By using the `static` command, you can specify a global address for each host. However, this manual binding can be too tedious to set up, or you might not have enough global addresses for every internal user. Dynamic address translation enables you to set up groups of internal addresses that can be assigned to a single global address or a pool of global addresses. The two main forms of dynamic address translations are NAT and PAT.

Network Address Translation

NAT enables you to dynamically assign groups of internal addresses to a pool of global addresses. This configuration takes a minimum of two commands: `nat` and `global`. Listing 5.3 demonstrates these commands.

Listing 5.3 The nat and global Commands Used for Dynamic Mappings

```
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)#
Pixfirewall(config)# route outside 0.0.0.0 0.0.0.0 169.254.8.254 1
Pixfirewall(config)#
Pixfirewall(config)# nat(inside) 1 0.0.0.0 0.0.0.0
```

Listing 5.3 The nat and global Commands Used for Dynamic Mappings *(continued)*

```
Pixfirewall(config)# global (outside) 1 169.254.8.2-169.254.8.6
               netmask 255.255.255.0
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

The `nat` and `global` commands work together to allow all traffic on the inside interface to exit the outside interface within the range of 169.254.8.2—169.254.8.6.

The 0.0.0.0 0.0.0.0 in the `nat` command is similar to a default route command; all IP addresses will use the global address range using a NAT ID of `1`. The first address will consume the IP address of 169.254.8.2, and the next will use 169.254.8.3, and so on. If a computer stops passing traffic through the firewall, the PIX will release the xlate slot and allow another computer to reuse the global address.

TIP

Use the **no nat** and **no global** commands to delete all **nat** and **global** configuration entries.

In the example in Figure 5.6, computers coming from network 192.168.1.0 use the global range of 169.254.8.2–169.254.8.6 and computers from network 10.0.0.0 use the range of 169.254.8.7–169.254.8.11. The NAT ID portion of the `nat` and `global` commands helps to bind which inside NAT pool and which global pools go together.

Listing 5.4 demonstrates the commands necessary to create the mappings in Figure 5.6.

Listing 5.4 nat and global Commands for Two Dynamic Mappings

```
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)#
Pixfirewall(config)# route outside 0.0.0.0 0.0.0.0 169.254.8.254 1
Pixfirewall(config)#
Pixfirewall(config)# nat (inside) 2 192.168.1.0 255.255.255.0
Pixfirewall(config)# nat (inside) 3 10.0.0.0 255.0.0.0
Pixfirewall(config)# global (outside) 2 169.254.8.2-169.254.8.6
               netmask 255.255.255.0
Pixfirewall(config)# global (outside) 3 169.254.8.7-169.254.8.11
               netmask 255.255.255.0
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

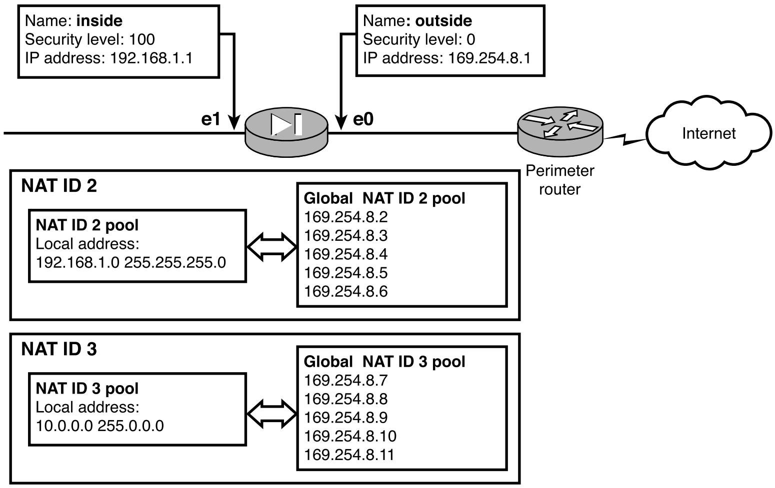

Figure 5.6 Using **nat** with two pools.

Port Address Translation

PAT enables you to use a single address for many internal addresses. For example, if your ISP assigns you only one address, you can have all your internal users share the same global IP address by implementing PAT. PAT modifies the port numbers of the source address to provide the tracking capability necessary on the return requests. For example, as Jack travels across the PIX, his address is changed from 192.168.1.11 port 1939 to the global address of 169.254.8.1 port number 5000. As Kristina travels across the PIX at the same time, her address of 192.168.1.25 port 1970 is changed to the same IP address Jack received—169.254.8.1—but with a different port number, such as 5001. The PIX tracks these IP address-to-port translations in the xlate table.

In Figure 5.7 all the inside addresses coming into the PIX will use the single address of 169.254.8.1 as their global source address. Because all the internal addresses use the same global address, PAT will also modify the source port to provide a specific mapping back to the internal source.

Listing 5.5 displays the PAT being set up with the `global` command using the option called `interface`. This option sets the outside interface IP address of 169.254.8.1 as the global address.

Listing 5.5 PAT and global Command for Dynamic Mappings

```
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)#
```

Listing 5.5 PAT and global Command for Dynamic Mappings *(continued)*

```
Pixfirewall(config)# route outside 0.0.0.0 0.0.0.0 169.254.8.254 1
Pixfirewall(config)#
Pixfirewall(config)# nat (inside) 1 0.0.0.0 0.0.0.0
Pixfirewall(config)# global (outside) 1 interface
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

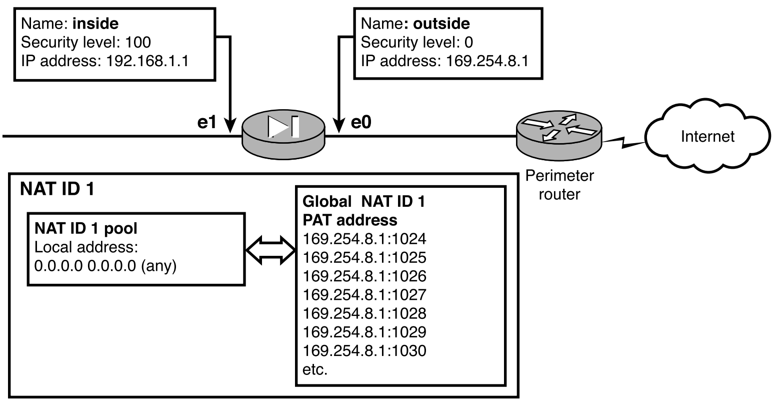

Figure 5.7 PAT example.

NAT and PAT Together

You've learned how to use NAT to translate internal addresses to a global address creating a one-to-one mapping. You have also seen how to use a single address and use PAT to change the source port numbers to create a mapping back to the internal source. Now, let's see how to use both at the same time.

Figure 5.8 and Listing 5.6 demonstrate the use of NAT and PAT together. Users from the inside first use the NAT address in the range of 169.254.8.1–169.254.8.5 until all the entries have been allocated. Then they begin to share the PAT address and use different port numbers.

Listing 5.6 NAT, PAT, and the global Command

```
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)#
Pixfirewall(config)# route outside 0.0.0.0 0.0.0.0 169.254.8.254 1
Pixfirewall(config)#
Pixfirewall(config)# nat (inside) 1 0.0.0.0 0.0.0.0
Pixfirewall(config)# global (outside) 1 169.254.8.2-169.254.8.6
               netmask 255.255.255.0
```

Listing 5.6 NAT, PAT, and the global Command *(continued)*

```
Pixfirewall(config)# global (outside) 1 169.254.8.11
              netmask 255.255.255.255
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

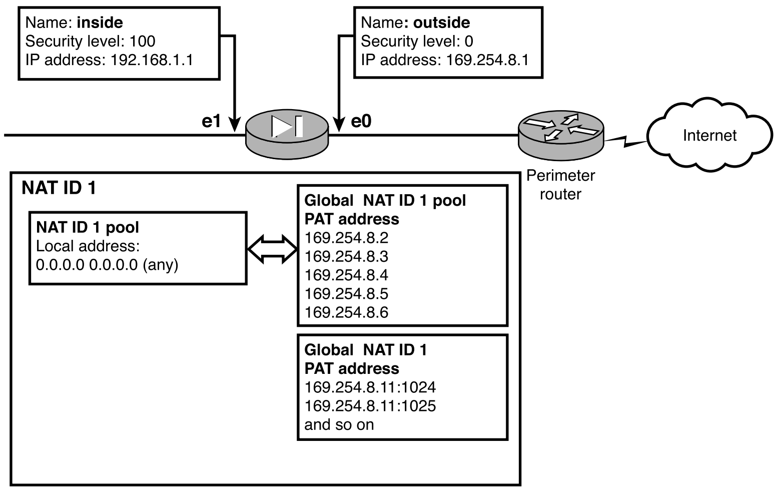

Figure 5.8 NAT and PAT together.

PAT has several problems when working with H.323, caching name servers, multimedia applications, and PPTP. So, if you are working with these technologies, use NAT instead of PAT.

Working Without NAT or PAT

You might encounter a situation in which addresses in the inside interface are just fine the way they are and don't need translating. If this is the case, you can use the `nat 0` command. For example, if your address is 192.168.1.11 and it doesn't need to be translated on the other side of the PIX, the `nat 0` command simply passes the traffic through without changing the source address.

nat 0

The `nat 0` command stands by itself. It doesn't need the use of a `global` command because it doesn't change the source address; it only passes it through the firewall like a router would.

In Figure 5.9 the `nat 0` command does not translate the three networks; it just passes them on.

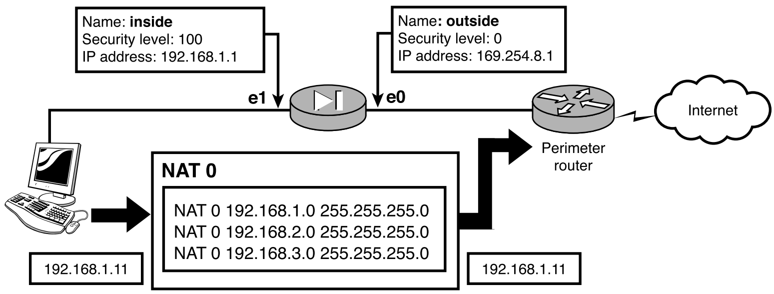

Figure 5.9 The **nat 0** command.

Listing 5.7 shows that three `nat 0` commands are used to allow the three subnets of 192.168.1.0, 192.168.2.0, and 192.168.3.0 to pass through the PIX firewall without translating their addresses.

Listing 5.7 Using the nat 0 Command

```
Pixfirewall(config)# nat (inside) 0 192.168.1.0 255.255.255.0
Pixfirewall(config)# nat (inside) 0 192.168.2.0 255.255.255.0
Pixfirewall(config)# nat (inside) 0 192.168.3.0 255.255.255.0
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

Using nat 0 with an Access Control List

You can accomplish the same `nat 0` task as in Listing 5.8 by using an access control list to point to a list of address you want to use. Listing 5.8 uses the `access-list` option in the `nat 0` command to point to an access list named `cool-no-nat`. The three `access-list` commands create the pool of addresses that will not be translated.

Beware that the mask on **access-list** commands on the PIX firewall are similar to normal subnet masks, not wildcard masks as on a Cisco IOS router. For example, to define a subnet for 192.168.1.0, the PIX would use a mask of 255.255.255.0, whereas IOS routers would use 0.0.0.255 for the same subnet.

A PIX firewall would use the following command:

`access-list 1 permit IP 192.168.1.0 255.255.255.0`

An IOS router, on the other hand, would use this command:

`access-list 1 permit IP 192.168.1.0 0.0.0.255`

Listing 5.8 Using the nat 0 Command with an Access List

```
Pixfirewall(config)# nat (inside) 0 access-list cool-no-nat
Pixfirewall(config)#
Pixfirewall(config)#access-list cool-no-nat permit IP 192.168.1.0
               255.255.255.0
Pixfirewall(config)#access-list cool-no-nat permit IP 192.168.2.0
               255.255.255.0
Pixfirewall(config)#access-list cool-no-nat permit IP 192.168.3.0
               255.255.255.0
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

Inbound Traffic

By default, the PIX prevents any traffic initiated on the lower security levels from accessing higher security level interfaces. However, when computers such as Web servers are located in a DMZ, you might need to open the firewall and allow external users to access the Web server. By using the `static` and `conduit` command, you can allow traffic initiated from the outside in to the DMZ or inside interfaces. Access lists can also be used to allow traffic in (they are covered in Chapter 6, “Access Control Lists and Traffic Control”).

For example, to allow Peter access to your Web server located in the DMZ, you first need to create a static mapping of the Web server to a global IP or port address. Then, you must use either the `conduit` command or an access list to allow traffic to pass from the lower security level interface to the DMZ.

You’ve seen the `static` command in action, and you know how to create a one-to-one mapping of IP addresses. But the `conduit` command is a special case because it is currently being replaced by the introduction of the access control list commands into the PIX IOS. Cisco doesn’t recommend using the `conduit` command, but you do need to be familiar with it. Using conduits enables you to control inbound access to the PIX. In Figure 5.10, any traffic from the outside is allowed to access the Web server.

Listing 5.9 demonstrates the `conduit` command.

Listing 5.9 The conduit Command

```
Pixfirewall(config)# static (dmz, outside) 169.254.8.2 172.16.0.5
Pixfirewall(config)# conduit permit tcp host 169.254.8.2 eq www any
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

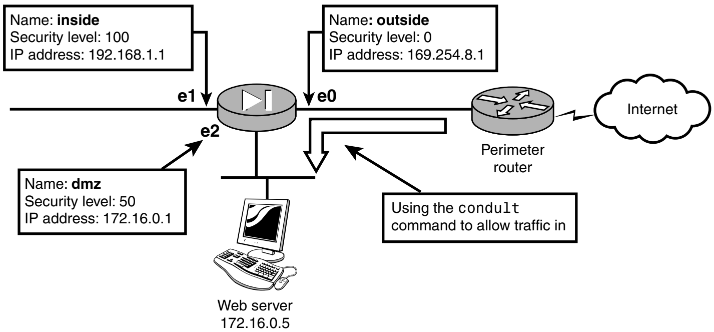

Figure 5.10 The **static** and **conduit** commands.

The `static` command in Listing 5.9 maps the Web server's internal address to the global address of 169.254.8.2. The `conduit` command permits any traffic destined to global address 169.254.8.2 with port 80 (www) to enter the PIX. Once inside, the PIX translates the destination of the global address of 169.254.8.2 to the mapped internal address of 172.16.0.5 and passes the traffic.

This setup works well when you have several NAT-able addresses. Figure 5.11 displays three servers in the DMZ that need to be accessed via the outside interface. The `static` commands map global addresses to DMZ addresses, and the conduits create exceptions to the ASA for certain protocols using the static mappings.

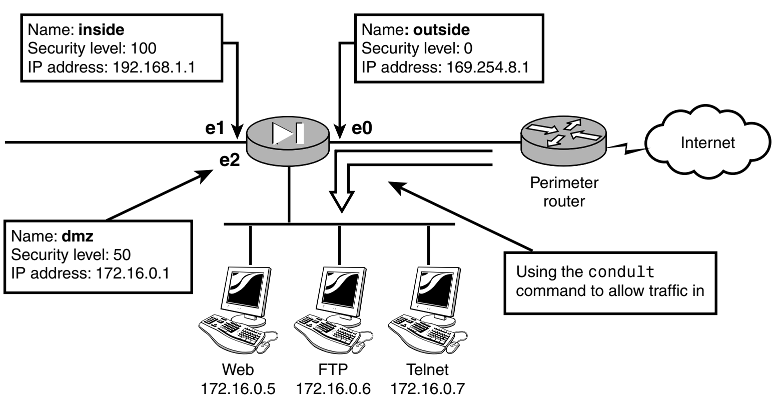

Figure 5.11 Static and conduit to multiple DMZ servers.

Listing 5.10 shows the `static` and `conduit` commands allowing traffic from any host to any of the three servers.

Listing 5.10 conduit Commands for Multiple DMZ Servers

```
Pixfirewall(config)# static (dmz, outside) 169.254.8.2 172.16.0.5
Pixfirewall(config)# static (dmz, outside) 169.254.8.3 172.16.0.6
Pixfirewall(config)# static (dmz, outside) 169.254.8.4 172.16.0.7
Pixfirewall(config)#
Pixfirewall(config)# conduit permit tcp host 169.254.8.2 eq www any
Pixfirewall(config)# conduit permit tcp host 169.254.8.3 eq ftp any
Pixfirewall(config)# conduit permit tcp host 169.254.8.4 eq telnet any
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

The **static** command is used to create binding and permanent mapping from an internal address to a global address. The **conduit** command is used to allow lower security level interfaces to access higher security level interfaces.

Port Redirection

Using NAT is great for creating one-to-one mappings of internal and global external addresses. However, if your ISP gives you only a single address to work with and you still want internal computers to be accessed via the Internet, port redirection is your solution. By using a single IP address on the outside, you can direct the traffic to the desired internal server by mapping the port numbers. For example, if Jack has only one outside address (169.254.8.1) but needs to access three servers via the Internet, he could create a mapping as shown in Table 5.4. Table 5.4 shows that external global addresses and ports are mapped to internal addresses and ports.

Table 5.4 Port Redirection Example

Internal Address: Port	External Global Address: Port
172.16.0.1:80	169.254.8.1:80
172.16.0.2:21	169.254.8.1:21
172.16.0.3:23	169.254.8.1:23

Whenever traffic comes in entering 169.254.8.1:80, it is redirected to the internal server at 172.16.0.1:80. This enables Jack to use a single IP address and still access several services hosted behind the firewall. Listing 5.11 is an example of using the `static` command with the `port` option to make one-to-one mappings using a single address and a specific port.

Listing 5.11 Port Redirection

```
Pixfirewall(config)# static (dmz, outside) 169.254.8.1 80 172.16.0.1 80
Pixfirewall(config)# static (dmz, outside) 169.254.8.1 21 172.16.0.2 21
Pixfirewall(config)# static (dmz, outside) 169.254.8.1 23 172.16.0.3 23
Pixfirewall(config)#
Pixfirewall(config)# conduit permit tcp host 169.254.8.1 eq www any
Pixfirewall(config)# conduit permit tcp host 169.254.8.1 eq ftp any
Pixfirewall(config)# conduit permit tcp host 169.254.8.1 eq telnet any
Pixfirewall(config)#
Pixfirewall(config)# exit
Pixfirewall# clear xlate
```

NOTE

In PIX OS version 6.0, the **static** command has been modified to allow port redirection without the need for a **conduit** command (see Cisco's site for more information: **http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_tech_note09186a0080094aad.shtml#topic9**).