Chapter 3

PIX Firewall Operations

Solutions in this chapter:

- Security Contexts
- The Bare Minimum: Outbound Traffic
- Opening Your Network: Allowing Inbound Traffic
- Outbound ACLs (New)
- Time-Based ACLs (New)
- NAT Control (New)
- Bypassing NAT
- Policy NAT
- Object Grouping

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Once the Cisco PIX firewall has been unboxed, plugged in, booted up, and configured with its initial system parameters, the first thing most security professionals want to do is configure it to pass traffic appropriately (i.e., according to the organization security policy). A firewall would not serve any purpose if it indiscriminately blocked all traffic. To properly protect a network environment, network traffic must be filtered in both outbound and inbound directions. The key to configuring a firewall is to permit only the traffic you want and block the traffic you do not want. This concept is easy to understand, but not always an easy task.

This chapter provides the basics needed to pass traffic through Cisco PIX firewalls. Perhaps one of the most important fundamentals to traffic passing is address translation, of which there are two types: static and dynamic. Once translation has been configured, the PIX will automatically allow all connections from a higher security interface to a lower security interface and deny all connections from a lower security interface to a higher security interface. To configure more granular access, you can permit or deny specific traffic using access lists.

The decisions to permit or deny specific traffic compose the firewall rules, typically in the form of access lists. Whether you are configuring rules for outbound or inbound traffic, the process is the generally the same:

1. Configure address translation.
2. Define an access list and apply it to an interface.

You must ensure that users can access the required network services through the firewall. You must also ensure that external services are available to one or more communities of users. While the process for filtering inbound and outbound traffic is the same, task details differ.

The Cisco PIX firewall offers several features that are valuable in managing traffic, including:

- Object Grouping simplifies access list configuration and maintenance
- TurboACLs
- Time-based ACLs
- Access list logging
- Enabling/disabling ACL entries
- NAT control
- Policy NAT

Throughout the chapter, we provide examples to describe the various commands. We also include a complex case study to reinforce concepts.

Security Contexts

Have you ever wished you could clone your Cisco PIX firewall? Ever had the need to have two very different security policies in place on your network, say one for the finance folks (very restricted) and one for the IT folks (anything goes)? Well, Cisco has listened, and for version 7.0, implemented something called security contexts. When you set up a security context, each context has its own security policies, interfaces, and supported features. This means that not all PIX firewall features are supported in security contexts. Some that are not supported when you have multiple security contexts include:

- Dynamic routing protocols
- VPN
- Multicast

When you start the PIX in single context mode and convert to multiple context mode, a new file called admin.cfg is created on the built-in flash. This is the default administrator security context. You can store multiple security contexts on the same flash or you can have the PIX download them from the network using TFTP, FTP, or HTTP(s).

NOTE

When converting from single security context mode to multiple security context mode, the original startup configuration is *not* saved, so always make a backup when working with security contexts. The running configuration is used to make the two new security context files.

Use the *mode* command to place the Cisco PIX firewall in multiple security context mode. Our options for the *mode* command are:

```
PIX1(config)# mode ?

configure mode commands/options:
  multiple    Multiple mode; mode with security contexts
  noconfirm   Do not prompt for confirmation
  single      Single mode; mode without security contexts
PIX1(config)#
```

To go from single mode to multimode:

```
PIX1(config)# mode multiple
WARNING: This command will change the behavior of the device
WARNING: This command will initiate a Reboot
```

```
Proceed with change mode? [confirm]
Convert the system configuration? [confirm]
WARNING: This command will initiate a Reboot
Proceed with change mode? [confirm]
Convert the system configuration? [confirm]
!!
The old running configuration file will be written to flash

The admin context configuration will be written to flash

The new running configuration f



***
*** --- SHUTDOWN NOW ---
***
*** Message to all terminals:
***
***   change mode
file was written to flash
Security context mode: multiple




Rebooting...
```

When you confirm, the Cisco PIX will reboot itself to enable the new mode. We can confirm the mode by using the *show* command:

```
PIX1# show mode
Security context mode: multiple
PIX1#
```

To restore the Cisco PIX to single mode security context, we need to copy the original (you did make a backup, right?) to flash:

```
PIX1(config)# copy flash:old_running.cfg startup-config
```

Then we will set the mode back to single:

```
PIX1(config)# mode single
WARNING: This command will change the behavior of the device
WARNING: This command will initiate a Reboot
Proceed with change mode? [confirm]
```

The Bare Minimum: Outbound Traffic

After completing the initial configuration, a primary task is allowing outbound traffic (such as from inside to outside). Outbound connections are from a higher security interface (e.g., the organization's internal network) to a lower security interface (e.g., an external network such as the Internet). This requires either configuring address translation or explicitly disabling it. Once address translation is configured, by default, if no access lists or apply/outbound statements are applied, all outbound traffic is allowed. This is a primary feature of the Adaptive Security Algorithm (ASA) and is the reason why security levels are so critical. Since the PIX is stateful, when an outbound connection is initiated, return traffic that is part of an established connection is allowed from the lower security interface to the higher security interface.

The *specific* approach for controlling outbound traffic consists of:

- Configuring *dynamic* address translation.
- Defining an access list and applying it to an interface on the PIX (optional).

NOTE

With version 7.0, the requirement for address translation policies to be in place before allowing network traffic to flow from an inside host to an outside destination has been eliminated, thanks to the NAT Control feature. For new configurations (i.e., a new installation of a PIX using v7.0), the translation rules are not required, and NAT Control is automatically disabled via the *no nat-control* command. For upgraded configurations (i.e., existing PIX firewall being upgraded to v7.0), the translation rules are required to preserve the functionality already defined in the configuration, and NAT Control is automatically enabled via the *nat-control* command. Note that NAT Control is different from identity NAT (nat 0). See the sections titled "NAT Control," "NAT Control (New)," and "Identity NAT."

Configuring Dynamic Address Translation

Address translation is the first requirement for passing outbound traffic. Address translation (through NAT and/or PAT) maps local IP addresses to global IP addresses. A local IP address is one from a network that is being protected by the PIX (e.g., your internal network), and is frequently a private, nonroutable IP address. A global IP address is one from a network on an outside interface of the PIX, and is frequently a public, routable IP address. The address translation configuration causes the PIX to translate the local IP address to the global IP address by making the appropriate substitution within the packet. Once NAT and/or PAT are configured, the PIX automatically allows traffic to traverse from a higher security interface to a lower security interface on the PIX firewall (also known as *outbound connections*). The PIX also permits any return traffic related to these outbound connections.

Configuration of NAT/PAT is a two-step process:

1. Use the *nat* command to identify the local addresses that will be translated.
2. Use the *global* command to define the global addresses to translate to.

NOTE

Address translation records are known as *translation slots* (or *xlate*) and are stored in a table known as the *translation table*. To view the contents of this table, use the *show xlate* command. The *xlate* timer monitors the translation table and removes records that have been idle longer than the defined timeout. By default, this timeout is set to three hours, and the current settings can be set using the *timeout xlate* command and can be verified by using the *show running-config timeout xlate* command.

The syntax of the *nat* command is as follows:

```
nat (real_ifc) nat_id real_ip [mask [dns] [outside] [[tcp] tcp_max_conns
[emb_limit]
[norandomseq]]] [udp udp_max_conns]
```

The keywords and parameters for the *nat* command are described here.

The *real_ifc* parameter is the interface that is the source of the traffic to be translated. It must match the name associated with this interface via the *nameif* command. If this parameter is not specified, the inside interface is assumed.

The *nat_id* parameter is an integer between 0 and 65,535 that establishes a mapping between the local IP addresses (*real_ip*) identified by the *nat* command and the global IP addresses specified by the *global* command. The *id* 0 is special and is used to specify that you do not want the specified local addresses translated: local addresses and global addresses are the same.

The *mask* parameter is used with real_ip to specify the IP addresses to be translated. The optional *dns* keyword translates the IP address included in DNS responses using active entries in the translation table. The optional *outside* keyword allows for external addresses to be translated.

The optional *tcp* keyword configures several TCP-related parameters. tcp_max_conns defines how many total concurrent TCP active connections are allowed, while emb_limit specifies how many concurrent half-open TCP connections are allowed. The default for both is 0, meaning unlimited connections. Too many half-open connections can be the result of a denial-of-service (DoS) attack, which tuning the emb_limit can help minimize.

By default, when performing address translation, the PIX firewall also randomizes the sequence numbers in TCP segments. The optional *norandomseq* keyword disables this randomization, which can be useful (and possibly necessary) when performing address translation twice (e.g., when you have two PIX firewalls in the path), and multiple randomization

is not desired. The *timeout* parameter defines how long an entry in the translation table may be idle.

The optional *udp* keyword configures a UDP-related parameter. The udp_max_conns parameter defines how many total concurrent active UDP "connections" are allowed.

NOTE

Because of its stateless nature, there is no such thing as a UDP "connection." The PIX firewall is stateful, and can permit return traffic in response to a UDP datagram. The stateless nature of UDP is also the reason why there is not a corresponding *emb_limit* for the *udp* keyword as there was with the *tcp* keyword. There is no such thing as a UDP half-open connection.

Once you have identified the local addresses to be translated using the *nat* command, you then need to specify the global addresses to which the local addresses should be translated. Use the *global* command to accomplish this:

```
global (mapped_ifc) nat_id {mapped_ip [-mapped_ip] [netmask mapped_mask]} |
interface
```

The mapped_ifc parameter defines the egress interface for outbound traffic; the default assumption is the outside interface.

The nat_id parameter pairs one or more nat statements to a global statement.

The mapped_ip parameter defines the global IP addresses for translation. If a single IP address is specified, port address translation is performed. If a range is specified (via -mapped_ip), network address translation is used until no more global addresses are available. Once the global addresses pool is exhausted, port address translation is performed.

The *netmask* keyword is associated with the mapped_ip range to specify the network mask. This determines the range of valid global addresses for the PIX to use, and ensures that the PIX does not use broadcast or network addresses in its translation.

If the global IP address to be used is assigned to an interface (e.g., the outside interface of the PIX), the *interface* keyword can be used instead of the mapped_ip parameter to specify this.

The use of the *nat* and *global* commands can be illustrated using an example organization. The fictitious Secure Corporation needs to network three locations and provide Internet access to its employees. It does not own any public IP addresses, and must use RFC 1918 private addresses for its internal networks. RFC 1918 addresses are not routable or usable on a public network such as the Internet. Secure Corporation is using private addresses because it does not want to re-address if it switches service providers. By using a private IP address scheme, the company can change public IP addresses whenever circumstances require, and all it will have to do is associate the new IP address range to the private IP addresses. Figure 3.1 shows the network layout. (Note: Even though it is a private address range, the 192.168.0.0/16 network is being used to represent the public IP address space in this chapter. Keep this in mind as you read the rest of the chapter.)

Figure 3.1 Network Address Translation

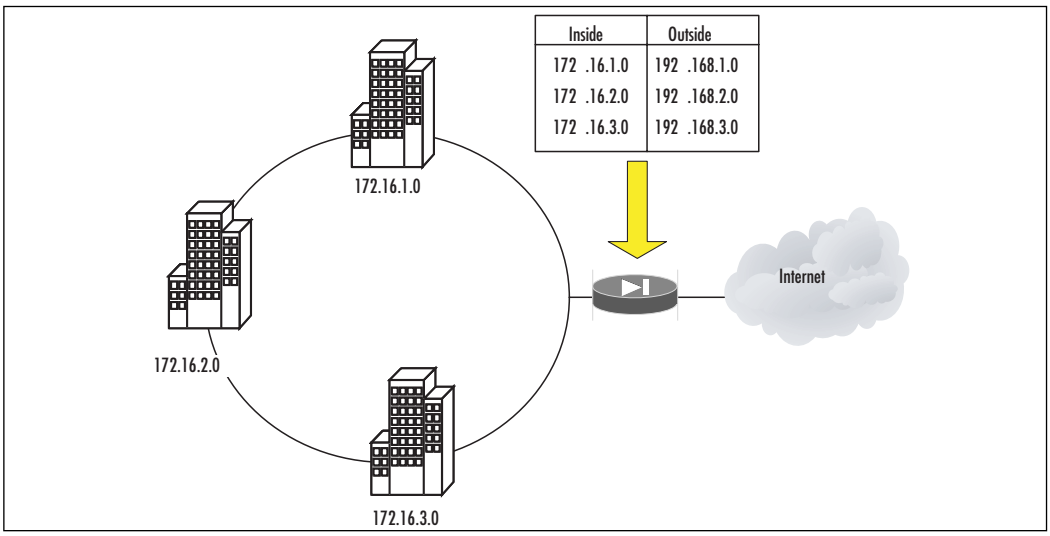


Figure 3.1 shows that each location has been assigned a 24-bit network from a range specified in RFC 1918. These ranges are 172.16.1.0/24, 172.16.2.0/24, and 172.16.3.0/24, respectively. The service provider has allocated a 24-bit subnet (192.168.1.0, 192.168.2.0, and 192.168.3.0) to each location, which needs to be mapped to a private address range. The following configuration allows each node to have a unique public IP address dynamically mapped from a pool established for each location. Traffic to be translated is identified using the *nat* command and then mapped to a pool of public IP addresses defined by the *global* command.

```
PIX1(config)# nat (inside) 1 172.16.1.0 255.255.255.0
PIX1(config)# global 1 192.168.1.1-192.168.1.254 netmask 255.255.255.0
PIX1(config)# nat (inside) 2 172.16.2.0 255.255.255.0
PIX1(config)# global 2 192.168.2.1-192.168.2.254 netmask 255.255.255.0
PIX1(config)# nat (inside) 3 172.16.3.0 255.255.255.0
PIX1(config)# global 3 192.168.3.1-192.168.3.254 netmask 255.255.255.0
PIX1(config)# exit
PIX1# clear xlate
```

NOTE

The *clear xlate* command is used to clear contents in the translation table. This command should be executed after any translation configuration changes are made; otherwise, there is a danger of stale entries sticking around in the translation table. However, note that this will also disconnect all current connections that use the translations.

Validate your configuration using the *show running-config nat* and *show running-config global* commands:

```
PIX1# show running-config nat
nat (inside) 1 192.168.1.0 255.255.255.0 0 0
nat (inside) 2 192.168.1.0 255.255.255.0 0 0
nat (inside) 3 192.168.1.0 255.255.255.0 0 0
PIX1# show running-config global
global (outside) 1 10.1.1.1-10.1.1.254 netmask 255.255.255.0
global (outside) 2 10.1.2.1-10.1.2.254 netmask 255.255.255.0
global (outside) 3 10.1.3.1-10.1.3.254 netmask 255.255.255.0
```

In this simple but unrealistic example, the provider allocated enough public addresses to allow one-to-one mappings between local and global addresses. What would happen if the provider did not allocate enough public addresses? Let's modify our example to where the provider only gave Secure Corp. a single 24-bit public address range (10.1.1.0/24). Instead of separate global pools for each location, there is one global pool for all to share, meaning that PAT is needed. PAT allows many IP addresses to be translated to a single IP address by incorporating both the IP address and the source port. The configuration would be as follows:

```
PIX1(config)# nat (inside) 1 192.168.1.0 255.255.255.0
PIX1(config)# nat (inside) 1 192.168.2.0 255.255.255.0
PIX1(config)# nat (inside) 1 192.168.3.0 255.255.255.0
PIX1(config)# global (outside) 1 10.1.1.1-10.1.1.254 netmask 255.255.255.0
PIX1(config)# exit
PIX1# clear xlate
```

NOTE

PAT works with DNS, FTP, HTTP, mail, RPC, RSH, Telnet, URL filtering, and outbound *traceroute*. PAT does not work with H.323, caching name servers, and PPTP.

To enable NAT on multiple interfaces, separate *global* commands are needed for each interface. The key is the same *id* on all the *global* commands to permit one set of *nat* commands on the translated interfaces to map a private IP address to one of many different global address ranges based on destination. For example, the following commands configure the PIX to translate the 192.168.1.0/24 network to either a 10.1.1.0/24 address or PAT to the DMZ interface IP address, depending on the interface the packet was going to exit:

```
PIX1(config)# nat (inside) 1 192.168.1.0 255.255.255.0
PIX1(config)# global (outside) 1 10.1.1.1-10.1.1.254 netmask 255.255.255.0
PIX1(config)# global (dmz) 1 interface
```

```
PIX1(config)# exit
PIX1# clear xlate
```

As with most commands on the PIX firewall, use the *no* keyword with the *nat* and *global* commands to remove them from the configuration.

Blocking Outbound Traffic (Defining an Access List)

Without any additional configuration, the PIX allows all higher security interfaces to send traffic to lower security interfaces. If you want to block some outbound traffic, it must be *explicitly* blocked. Controlling the outbound traffic that is allowed to traverse the PIX firewall is always a part of a well-designed security policy. With version 7.0, there is only one way to accomplish this task: using access lists.

NOTE

In previous PIX software versions, you could also block outbound traffic using the *outbound* command. With version 7.0, this command has been completely replaced by the *access-list* command, and is no longer supported. Cisco has been discouraging the use of the *outbound* command, so this should not be a surprise. Existing *outbound* commands are not automatically converted to *access-list* commands. Refer to the section "Conduit and Outbound" for more information, and for a description of how to convert *outbound* commands to *access-list* commands.

Access Lists

Access lists on the PIX firewall are very similar to those used on Cisco routers, and can be used to limit the traffic based on several criteria, including source address, destination address, source TCP/UDP ports, and destination TCP/UDP ports. Access list configuration is a two-step process:

1. Define the access list by creating permit and deny statements using the *access-list* command.
2. Apply the access list to an interface using the *access-group* command.

There are three different basic protocol classes for the *access-list* command: IP, TCP/UDP, and ICMP. Each of these classes has an *access-list* command syntax that differs slightly from the others, as shown here:

```
access-list id [line line-number] [extended] {deny | permit} protocol {host
sip | sip mask | any} {host dip | dip mask | any}
```

```
access-list id [line line-number] [extended] {deny | permit} {tcp | udp}
{host sip | sip mask | any} [operator port] {host dip | dip mask | any}
[operator port]access-list id [line line-number] [extended] {deny | permit}
icmp {host sip | sip mask | any} {host dip | dip mask | any} [icmp_type ]
```

The parameters and keywords that are common across all three *access-list* command syntaxes are identified and described in this section. Parameters and keywords that are specific to each of the command syntaxes are identified and described in the following paragraphs.

The *id* parameter identifies the access list and can be either a name or a number. The order of access-list statements is sequential from top to bottom. The first entry that matches is applied, and further processing is halted.

The *line* keyword and *line-num* parameter allow entries to be inserted at a specific location within the access list.

The *extended* keyword identifies this as an extended access list that can specify source and destination IP addresses and ports.

The source IP address is specified using the *sip* parameter and identifies the origin of the traffic.

The destination IP address is specified using the *dip* parameter and identifies the destination of the traffic.

The mask parameter specifies the netmask bits to apply to either *sip* or *dip*.

The *any* keyword specifies all networks or hosts, and is the equivalent of a network of 0.0.0.0 and a mask of 0.0.0.0.

The host keyword followed by an IP address specifies a single host.

NOTE

The syntax for access lists on the PIX firewall is very similar to that of IOS routers. The key difference is that access lists on PIX firewalls use standard wildcard masks, whereas on routers they use inverse wildcard masks. For example, when blocking a 24-bit subnet, you would use a mask of 255.255.255.0 on a PIX firewall and a mask of 0.0.0.255 on a Cisco router.

IP Protocol Access List Parameters and Keywords

In the IP protocol *access-list* command syntax, the *protocol* parameter specifies the IP protocol. You can either enter the numerical value or specify a literal name. Table 3.1 lists possible literal names.

Table 3.1 Literal Protocol Names and Values

Literal	Value	Description
ah	51	Authentication header for IPv6, RFC 1826
eigrp	88	Enhanced Interior Gateway Routing Protocol
esp	50	Encapsulated Security Payload for IPv6, RFC 1827
gre	47	General Routing Encapsulation
icmp	1	Internet Control Message Protocol, RFC 792
igmp	2	Internet Group Management Protocol, RFC 1112
igrp	9	Interior Gateway Routing Protocol
ip	0	Internet Protocol
ipinip	4	IP-in-IP encapsulation
nos	94	Network Operating System (Novell's NetWare)
ospf	89	Open Shortest Path First routing protocol, RFC 1247
pcp	108	Payload Compression Protocol
snp	109	Sitara Networks Protocol
tcp	6	Transmission Control Protocol, RFC 793
udp	17	User Datagram Protocol, RFC 768

TCP/UDP Protocol Access List Parameters and Keywords

In the TCP/UDP protocol *access-list* command syntax, the parameters and keywords are identified and described in this section.

The *tcp* and *udp* keywords specify whether this access list entry applies to TCP or UDP traffic.

The *operator* and *port* identify source and destination ports.

To specify all ports, do not specify an operator and port.

To specify a single port, use the *eq* keyword as the operator.

To specify all ports less than a specified port, use the *lt* keyword as the operator.

To specify all ports greater than a specified port, use the *gt* keyword as the operator.

To specify all ports except a specific one, use the *neq* keyword as the operator.

To specify a range of ports, use the *range* keyword as the operator.

The *port* can be specified using either a number or a literal name. A list of literal port names is presented in Table 3.2.

Table 3.2 Literal Port Names and Values

Name	Port	Protocol	Name	Port	Protocol	Name	Port	Protocol
bgp	179	tcp	http	80	tcp	radius	1645, 1646	udp
biff	512	udp	hostname	101	tcp	rip	520	udp
bootpc	68	udp	ident	113	tcp	smtp	25	tcp
bootps	67	udp	irc	194	tcp	snmp	161	udp
chargen	19	tcp	isakmp	500	udp	snmptrap	162	udp
citrix-ica	1494	tcp	klogin	543	tcp	sqlnet	1521	tcp
cmd	514	tcp	kshell	544	tcp	sunrpc	111	tcp/udp
daytime	13	tcp	login	513	tcp	syslog	514	udp
discard	9	tcp/udp	lpd	515	tcp	tacacs	49	tcp/udp
dnsix	195	udp	mobile-ip	434	udp	talk	517	tcp/udp
domain	53	tcp/udp	nameserver	42	udp	telnet	23	tcp
echo	7	tcp/udp	netbios-dgm	138	udp	tftp	69	udp
exec	512	tcp	netbios-ns	137	udp	time	37	udp
finger	79	tcp	nntp	119	tcp	uucp	540	tcp
ftp	21	tcp	ntp	123	udp	who	513	udp
ftp-data	20	tcp	pim-auto-rp	496	tcp/udp	whois	43	tcp
gopher	70	tcp	pop2	109	tcp	www	80	tcp
h323	1720	tcp	pop3	110	tcp	xdmcp	177	tcp

Note that the system-defined port mapping of *http* is the same as *www* and is silently translated in the configuration.

ICMP Access List Parameters and Keywords

In the ICMP *access-list* command syntax, the parameters and keywords are identified and described as follows:

- The *icmp* keyword applies this access list entry to ICMP traffic.
- The *icmp_type* parameter identifies the ICMP message type, and can be specified using either a number or a literal name. A list of ICMP message types and literal names can be found in Table 3.3.

Table 3.3 ICMP Message Types

ICMP Type	Literal
0	echo-reply
3	unreachable
4	source-quench
5	redirect
6	alternate-address
8	echo
9	router-advertisement
10	router-solicitation
11	time-exceeded
12	parameter-problem
13	timestamp-reply
14	timestamp-request
15	information-request
16	information-reply
17	mask-request
18	mask-reply
31	conversion-error
32	mobile-redirect

After configuring an access list, you must apply it to an interface using the following command: access-group *access-list* {in | out} interface *interface_name*. The parameters and keywords of the *access-group* command are identified and described here.

The *access-list* parameter specifies which access list statements to apply to the interface. This parameter value must correspond to the id (name) specified in previous access-list commands.

The *in* or *out* keyword is used to specify whether the access list is applied to packets that are inbound to the interface or outbound from the interface.

The *interface* keyword and *interface_name* parameter specify the interface to which the access-list statements should be applied.

Applying an access list to an interface via the *access-group* command denies or permits traffic as it enters the specified interface.

NOTE

In previous versions of the PIX software, access lists on the PIX firewall could only be applied to traffic *entering* an interface via the *in* keyword. In version 7.0, the access lists can also be applied to traffic as it exits an interface via the *out* keyword.

Access lists have an implicit *deny all* at the end. Unless traffic has been specifically permitted within the access list, it will be denied. You can create very complex access lists simply by following the flow of what should and should not be allowed. Only one access list at a time can be applied to an interface.

Let's now look at an example of Secure Corp., which has just purchased a new PIX firewall for its network in New York, as shown in Figure 3.2. All the servers at the site, as well as all the clients within the network, are located on the inside interface of the PIX. The site uses a single network with the address space of 192.168.0.0/24. The ISP has assigned the 10.1.1.0/24 public network to use.

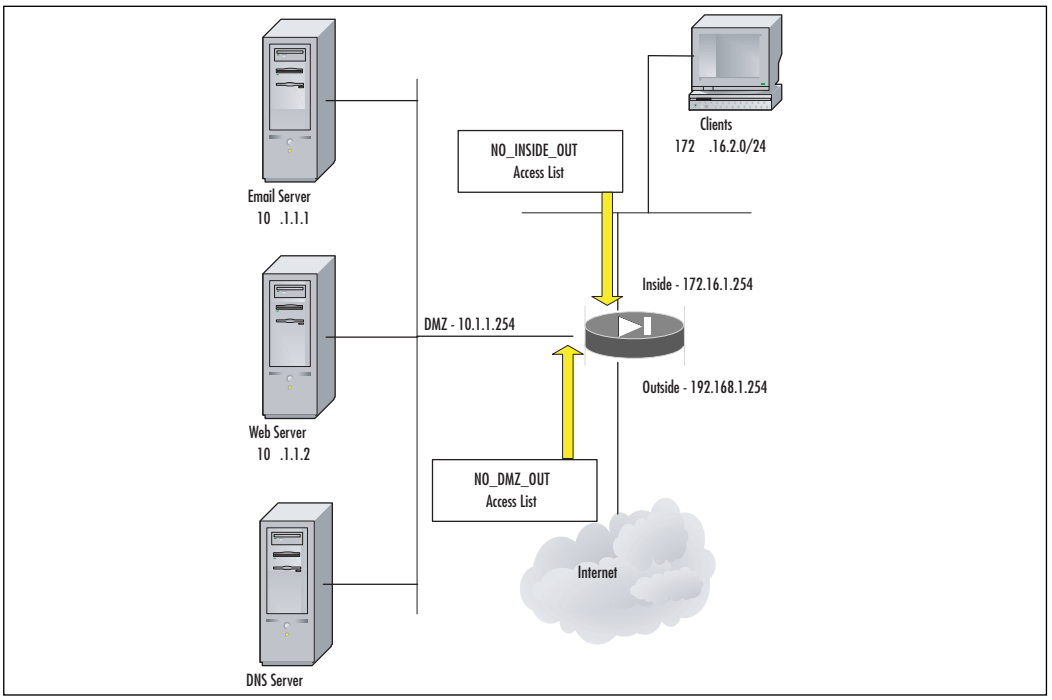
Internal clients should not be permitted to send certain traffic such as that associated with malware infection vectors. This includes Common Internet File System/Server Message Block (CIFS/SMB) traffic, a file sharing protocol. Other prohibited traffic includes Trivial File Transfer Protocol (TFTP), bootp, Simple Network Management Protocol (SNMP), SQL*Net, Kazaa, and P2P Networking traffic. Other than this explicitly prohibited traffic, the clients should have unrestricted access to the Internet.

DMZ servers should only have Internet access using protocols that support their core function. For example, the e-mail server should be permitted to send and receive Simple Mail Transfer Protocol (SMTP) traffic, the Web server should be permitted to send and receive only HyperText Transfer Protocol (HTTP) and HTTP over SSL (HTTPS) traffic, and the DNS server should be permitted to send and receive only Domain Name Service (DNS) traffic.

Since the DMZ interface is a higher security interface than the outside interface, any traffic initiated from the DMZ servers to the Internet will be implicitly permitted. We want to apply the same restrictions to the DMZ servers that have been applied to internal clients.

Because there should be no Web browsing from the servers, the company has defined a policy to prohibit outbound Web (i.e., HTTP and HTTPS) traffic from the DMZ servers.

Figure 3.2 An Outbound Access List



Secure Corporation's requirements are satisfied by two outbound access lists: one applied to the inside interface and one applied to the DMZ interface. The following commands define and apply the inside outbound access list:

```
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq 135
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq 135
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq netbios-ns
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq netbios-dgm
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq netbios-ssn
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq 139
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq 445
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq 445
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq tftp
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq bootpc
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq bootps
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq snmp
PIX1(config)# access-list NO_INSIDE_OUT deny udp any any eq snmptrap
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq sqlnet
```

```
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq 1214
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq 3408
PIX1(config)# access-list NO_INSIDE_OUT deny tcp any any eq 3531
PIX1(config)# access-list NO_INSIDE_OUT permit any any
PIX1(config)# access-group NO_INSIDE_OUT in interface inside
PIX1(config)# exit
```

The following commands define and apply the DMZ outbound access list, which differs from the inside outbound list primarily in the prohibition of Web traffic:

```
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq 135
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq 135
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq netbios-ns
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq netbios-dgm
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq netbios-ssn
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq 139
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq 445
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq 445
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq tftp
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq bootpc
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq bootps
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq snmp
PIX1(config)# access-list NO_DMZ_OUT deny udp any any eq snmptrap
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq sqlnet
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq 1214
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq 3408
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq 3531
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq www
PIX1(config)# access-list NO_DMZ_OUT deny tcp any any eq https
PIX1(config)# access-list NO_DMZ_OUT permit tcp10.1.1.1 any eq smtp
PIX1(config)# access-list NO_DMZ_OUT permit tcp10.1.1.3 any eq dns
PIX1(config)# access-list NO_DMZ_OUT permit udp10.1.1.3 any eq dns
PIX1(config)# access-group NO_DMZ_OUT in interface DMZ
PIX1(config)# exit
```

It is important to note that we have not yet covered how to configure inbound access. The preceding access lists only allow these servers to initiate contact with other servers—as a client would do. For example, the e-mail server can send mail to another domain, but it cannot receive it. The DNS server can resolve domain information from another domain, but it cannot respond to queries from other domains. In the section titled "Opening Your Network: Allowing Inbound Traffic," we cover in detail how inbound access is enabled.

One useful feature in configuring the PIX is the *name*, which maps a name alias to an IP address. When configuring, instead of referencing a host by its IP address, the host can be

referenced by a name. This can aid in configuring and troubleshooting complex configurations. It can also ease address changes: the name remains the name but the IP address can be changed without having to modify access lists. The syntax for the command is:

```
name ip_address name
```

For example, the following command maps the names *emailserver*, *webserver*, and *dnsserver* to the 172.16.1.1, 172.16.1.2, and 172.16.1.3 IP addresses, respectively:

```
PIX1(config)# name10.1.1.1 emailserver
PIX1(config)# name10.1.1.2 webserver
PIX1(config)# name10.1.1.3 dnsserver
```

The names *emailserver*, *webserver*, and *dnsswerver* can now be used in access lists instead of IP addresses.

Opening Your Network: Allowing Inbound Traffic

At some point, you will most likely face a requirement to enable untrusted and unknown hosts to initiate sessions with your trusted and protected devices such as servers. For example, users originating on the Internet may need to establish communications with your servers in the DMZ. The PIX would not be useful if it could not allow and control traffic from untrusted sources into networks containing critical systems, such as a corporate Web server. The PIX ASA treats inbound traffic (from lower security interface to a higher security interface) differently from outbound traffic.

Unlike outbound traffic, inbound traffic is denied by default. This ensures that the boundaries defined by the interface security levels are valid and not circumvented. As with outbound traffic, allowing inbound traffic is a two-step process. A static translation must be defined, and an access list be created to allow the inbound traffic.

NOTE

The *conduit* command has been completely superseded by access lists in 7.0. Cisco has been discouraging the use of the *conduit* command since 6.*x*, so this is not a big surprise.

Static Address Translation

When a publicly accessible server (hopefully located in a DMZ) is protected by a PIX firewall, connections initiated on a lower security interface to a higher security interface must explicitly be allowed. Allowing inbound traffic starts with the creation of a static address

translation. The *static* command permanently maps global-to-local IP addresses. The syntax for the command is:

```
static (real_ifc,mapped_ifc) {mapped_ip | interface} {real_ip [netmask
mask]} | {access-list access_list_name} [dns] [norandomseq [nailed]] [[tcp]
[max_conns [emb_lim]] [udp udp_max_conns]
```

The parameters and keywords of the *static* command are identified and described here.

The real_ifc parameter is the interface to which the server being translated is connected.

The mapped_ifc parameter is the interface of the mapped global IP address. This is the interface where you are making the device (e.g., server) visible and accessible.

The mapped_ip parameter is the global IP address that should be used for translation.

The real_ip parameter is the local IP address that should be translated. It is typically the actual, or real, IP address of the device (e.g., server) that you are making visible and accessible.

The netmask keyword and mask parameter are used when statically translating more than one IP address at a time.

The default value for max_cons, em_lim, and udp_max_conss is 0 (unlimited). Their meanings are the same as in the *nat* command.

In Figure 3.3, Secure Corp. has three servers connected to the DMZ network. The following *static* commands establish static address translation for these servers.

```
PIX1(config)# static (dmz, outside) 192.168.1.1 10.1.1.1 netmask
255.255.255.255 0 0
PIX1(config)# static (dmz, outside) 192.168.1.2 10.1.1.2 netmask
255.255.255.255 0 0
PIX1(config)# static (dmz, outside) 192.168.1.3 10.1.1.3 netmask
255.255.255.255 0 0
```

These commands provide the necessary translation to make the DMZ servers accessible via the 192.168.1.1, 192.168.1.2, and 192.168.1.3 addresses. With multiple DMZ servers, instead of configuring a separate static entry for each, you could configure a single *static* command with the appropriate netmask. For example, for 14 DMZ servers with the IP addresses of 10.1.1.1 through 10.1.1.15, you would use the following command:

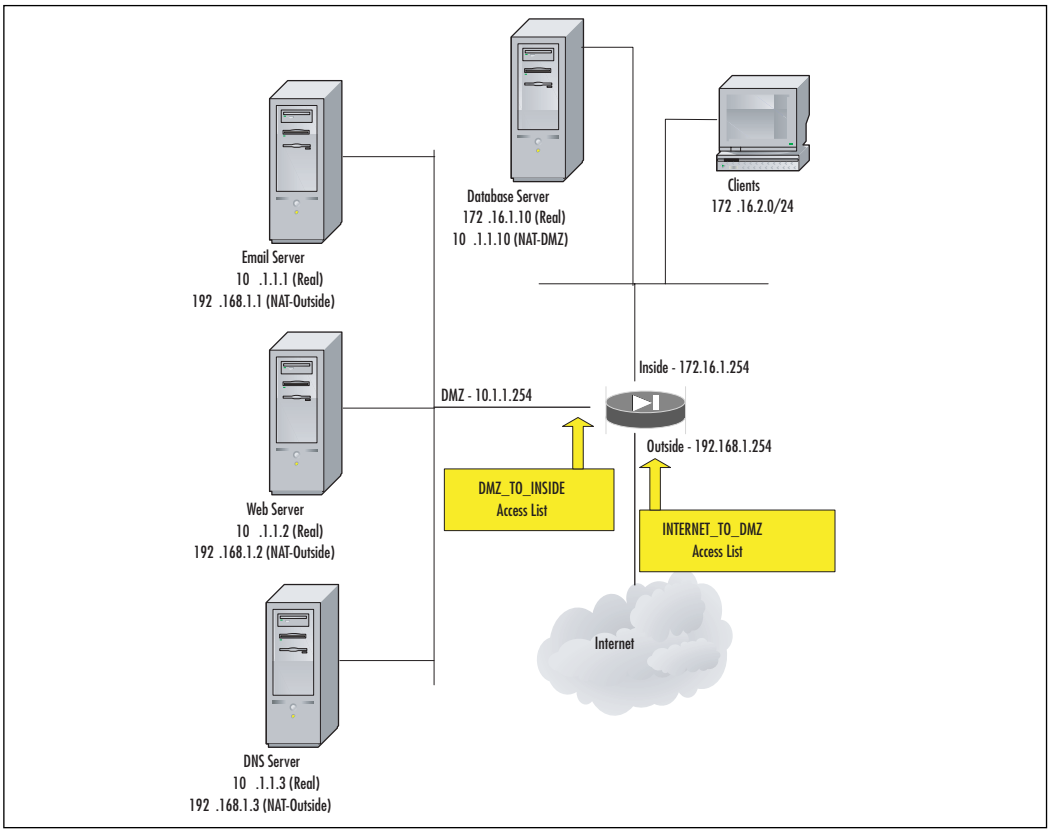
```
PIX1(config)# static (dmz, outside) 192.168.1.0 10.1.1.1.0 netmask
255.255.255.240 0 0
```

Now consider the fact that the Web server located in the DMZ needs to access a database server located on the inside interface of the PIX, as shown in Figure 3.3.

The process is the same: Whenever a lower security interface needs to access a higher security interface, a static translation needs to be created. The following configuration translates the real IP address of the internal database server (192.168.1.10) to an address accessible by the DMZ Web server (172.16.1.10):

```
PIX1(config)# static (inside, dmz) 10.1.1.10 172.168.1.10 netmask
255.255.255.255 0 0
```

Figure 3.3 Static Address Translation



Static translation alone is not enough to enable lower to high security communications; an access list must be defined to explicitly allow this traffic. The *static* command only creates a static address mapping between global and local IP addresses. Since the default action for inbound traffic is to drop it, the next step is to create an access list to allow the traffic to enter the PIX.

Access Lists

Creating an access list to allow inbound access is similar to creating an access list for outbound access. The command syntax is the same, as are all the parameters. The key difference is that static translation must be configured to enable lower to higher security traffic. For the Secure Corporation example in Figure 3.4, define and apply two access lists: one that permits communication from the Internet to the DMZ servers, and one that permits communication from the DMZ Web server to the internal database server.

The following commands define and apply the Internet inbound access list to the DMZ network:

```
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any 192.168.1.1 eq smtp
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any 192.168.1.2 eq web
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any 192.168.1.1 eq
https
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any 192.168.1.1 eq dns
PIX1(config)# access-list INTERNET_TO_DMZ permit udp any .1.1 eq dns
PIX1(config)# access-group INTERNET_TO_DMZ in Outside
PIX1(config)# exit
```

The following commands define and apply the DMZ inbound access list to the internal network:

```
PIX1(config)# access-list DMZ_TO_INSIDE permit tcp host 10.1.1.2  host
10.1.1.10 eq sqlnet
PIX1(config)# access-group DMZ_TO_INSIDE in interface DMZ
PIX1(config)# exit
```

Remember that there is an implicit "deny all" at the end of both access lists. This ensures that traffic not explicitly permitted to flow from a lower to higher security interface is denied. In our example, notice in the DMZ_TO_INSIDE access list that SQLNet traffic is only permitted between the DMZ Web server and the internal database server. Both the source and destination IP addresses are specified in the access-list statement. Only one access list (per direction) can be applied to an interface, and the DMZ_TO_INSIDE access list has to be combined with the NO_DMZ_OUT access list statements from the previous section to achieve the policy desired by Secure Corporation.

ICMP Access Lists

ICMP is a useful diagnostic protocol, perhaps best known and used via two useful utilities: ping and traceroute. Both tools generate ICMP messages, and use the responses to determine reachability and path information. Improperly controlled, ICMP can also be an attacker's most useful tool to pry open your organization. In addition, certain protocols may need ICMP to make path discoveries, or to determine reachability before session establishment. All of this combines to make troubleshooting a difficult issue. The lack of ICMP responses could indicate a network problem, or the firewall doing what it was created to do—protect your networks.

For this reason, the PIX firewall by default blocks ICMP, except as follows. Devices can ping the directly connected interfaces of the firewall appliance. Devices may ping each other, as long as they do not transit the firewall. ICMP traffic from the outside network to any higher security interface is blocked by default. Before you open your firewall to ICMP traffic, determine what traffic and between what networks you need to allow. If it has no value to you, do not allow it.

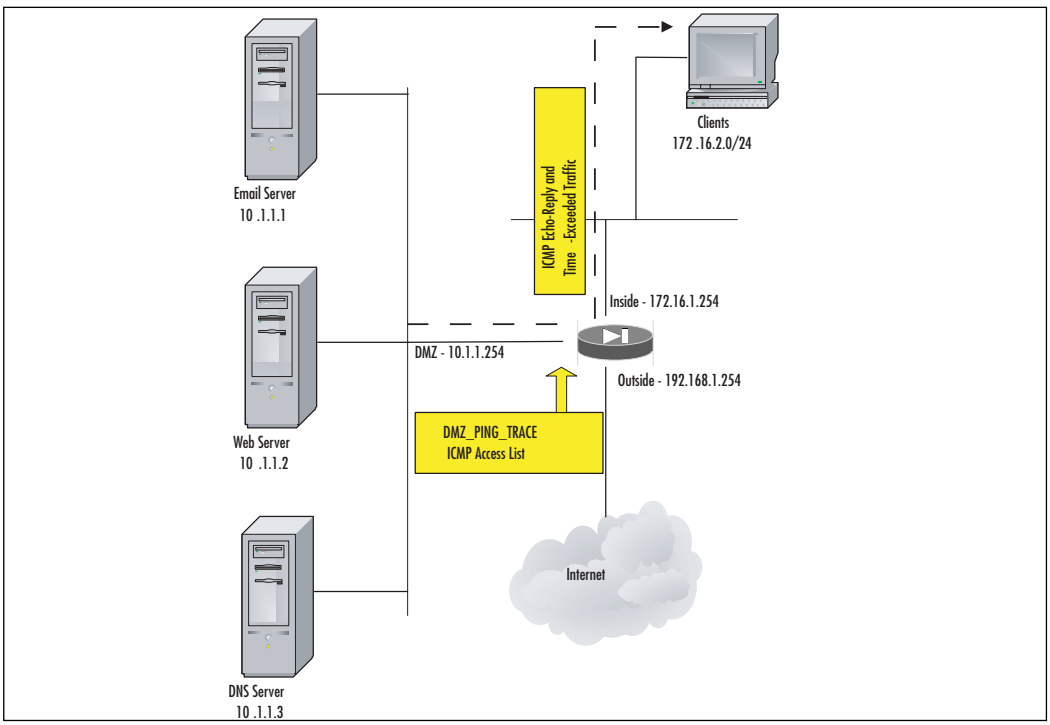
There are two approaches for permitting ICMP traffic to traverse the PIX:

- **Access lists** Because ICMP is a connectionless protocol, you need access lists to allow ICMP in both directions (by applying access lists to the source and destination interfaces).

- **ICMP inspection engine** You need to enable the ICMP inspection engine, which treats ICMP conversations as stateful connections. The ICMP inspection engine uses the Modular Policy Framework (MPF), a new PIX software v7.0 feature.

In this chapter we will show you how to use access lists to permit ICMP traffic to traverse the PIX. In Figure 3.4, we want the devices on the inside network to be able to ping and traceroute to devices on the DMZ network.

Figure 3.4 ICMP ACL



By default, the ICMP traffic from the inside network to the DMZ network will be permitted; therefore, we do not need to apply an access list to the inside interface. To allow the return traffic from the DMZ, we need to permit the following:

- For pings from the inside to the DMZ, the ICMP Echo packets from the inside to the outside will be permitted by default; however, we need to allow ICMP Echo Reply packets from the DMZ to the inside.
- For traceroute from the inside to the DMZ, the packets from the inside to the outside will be permitted by default; however, we need to allow ICMP Time Exceeded packets from the DMZ to the inside.

The following commands will accomplish this:

```
PIX1(config)# access-list DMZ_PING_TRACE permit icmp 10.1.1.0 255.255.255.0
172.16.0.0 255.255.240.0 eq echo-reply
PIX1(config)# access-list DMZ_PING_TRACE permit icmp 10.1.1.0 255.255.255.0
172.16.0.0 255.255.240.0 eq time-exceeded
PIX1(config)# access-group DMZ_PING_TRACE in DMZ
PIX1(config)# exit
```

Port Redirection

Port redirection allows one public IP address to serve as the public IP address for more than one server. Port redirection allows you to define a mapping between a port on a public IP address and a port on a private IP address. To enable redirection, an access list still must be created, since the traffic is traversing from a lower security interface to a higher security interface.

Because the mapping can be set at the port level, one IP address can serve as the gateway to many servers behind the PIX. For example, Secure Corp. has set up a network at its Toronto site and has been assigned only one public IP address from the ISP. At this site, Secure Corp. has two Web servers, one Telnet server, and one FTP server. How can it make all these services accessible publicly with a single IP address? By using the *static* command to perform port redirection:

```
static (real_ifc,mapped_ifc) {tcp | udp} {mapped_ip | interface}
mapped_port {real_ip real_port [netmask mask]} | {access-list
access_list_name} [dns] [norandomseq [nailed]] [[tcp]
[max_conns [emb_lim]] [udp udp_max_conns]
```

We discussed the *static* command earlier in the chapter, so we will not go through all the parameters again. However, we introduced some new parameters in the preceding section:

- The *tcp* and *udp* keywords are used to specify TCP or UDP port redirection through static PAT.
- The *mapped_port* and *real_port* parameters specify the mapped port (i.e., external port accessible outside the PIX) and the real port (i.e., actual port listening on the server), respectively.
- Instead of using the *mapped_ip* parameter, you can use the *interface* keyword to specify the IP address of the PIX interface specified in the *mapped_ifc* parameter. This option is important if you do not have any additional usable public IP addresses.

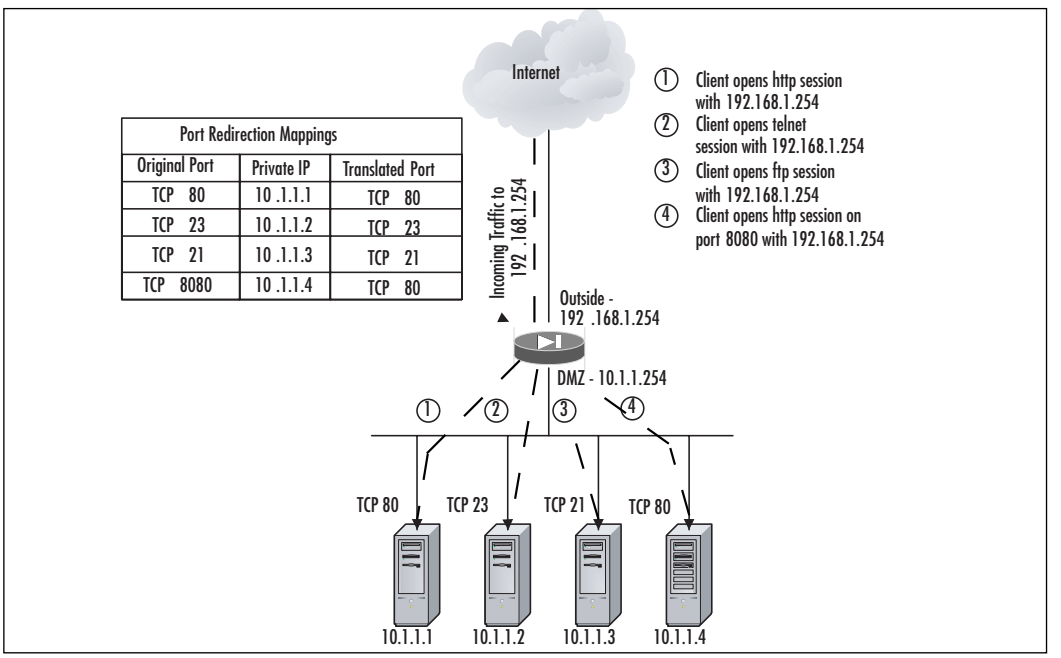
To configure port redirection for the first Web server using the PIX public IP address as the Web server's public address:

```
PIX1(config)# static (dmz, outside) tcp interface 8010.1.1.1 80
```

If the company also wanted to host Telnet, FTP, and another Web server, three more *static* commands would have to be added to map the global ports to the correct servers. Since the Web port is already taken, a high port (8080) is chosen for access to the second Web server. This example is shown in Figure 3.5. The additional commands are as follows:

```
PIX1(config)# static (dmz, outside) tcp interface 2310.1.1.2 23
PIX1(config)# static (dmz, outside) tcp interface 21 10.1.1.3 21
PIX1(config)# static (dmz, outside) tcp interface 8080 10.1.1.4 80
```

Figure 3.5 Port Redirection



Enabling/Disabling of ACL Entries (New)

PIX software version 7.0 has introduced the capability to temporarily disable an access control entry without removing it from the configuration file. This is a powerful troubleshooting tool for testing and fine-tuning access control lists. If you are troubleshooting some communication issues through the PIX firewall and are not sure which access control entry is causing the problem, you can selectively disable an entry by including the **inactive** keyword in the appropriate **access-list** statement. Similarly, if you want to temporarily disable an entry that may be re-activated at some point in the future, include the **inactive** keyword. For example, to disable the access control list entry permitting inbound Web traffic to the DMZ Web server:

```
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any host 10.1.1.2 eq
web inactive
PIX1(config)# exit
```

Outbound ACLs (New)

Access lists can be used to filter risky outbound traffic such as CIFS, TFTP, bootp, SNMP, SQL*Net, Kazaa, and P2P Networking traffic. In an earlier example, we created and applied access lists to the internal and DMZ interfaces to block this traffic to the Internet. This leads into our section on a new PIX software version 7.0 feature called "Outbound ACLs." In previous PIX OS versions, access lists could be applied to only packets that were *inbound* to the *interface*. With the PIX v7.0 Outbound ACL feature, access lists can be applied to either *inbound to* or *outbound from* the interface. However, you can apply only one ACL per direction on each interface.

NOTE

It is important to differentiate the terms "inbound" and "outbound" within the context of Outbound ACLs from when they are used to describe traffic flow within the context of perimeter network traversal. For Outbound ACLs, "inbound" and "outbound" refer to the application of an access list on an interface, either to traffic entering the security appliance on an interface or traffic exiting the security appliance on an interface. For perimeter network traversal, "inbound" refers to the movement of traffic from a lower security interface to a higher security interface, while "outbound" refers to higher to lower interface traffic.

To implement an Outbound ACL, use the *access-group* command with the *out* keyword instead of the *in* keyword:

```
access-group access-list out interface interface_name
```

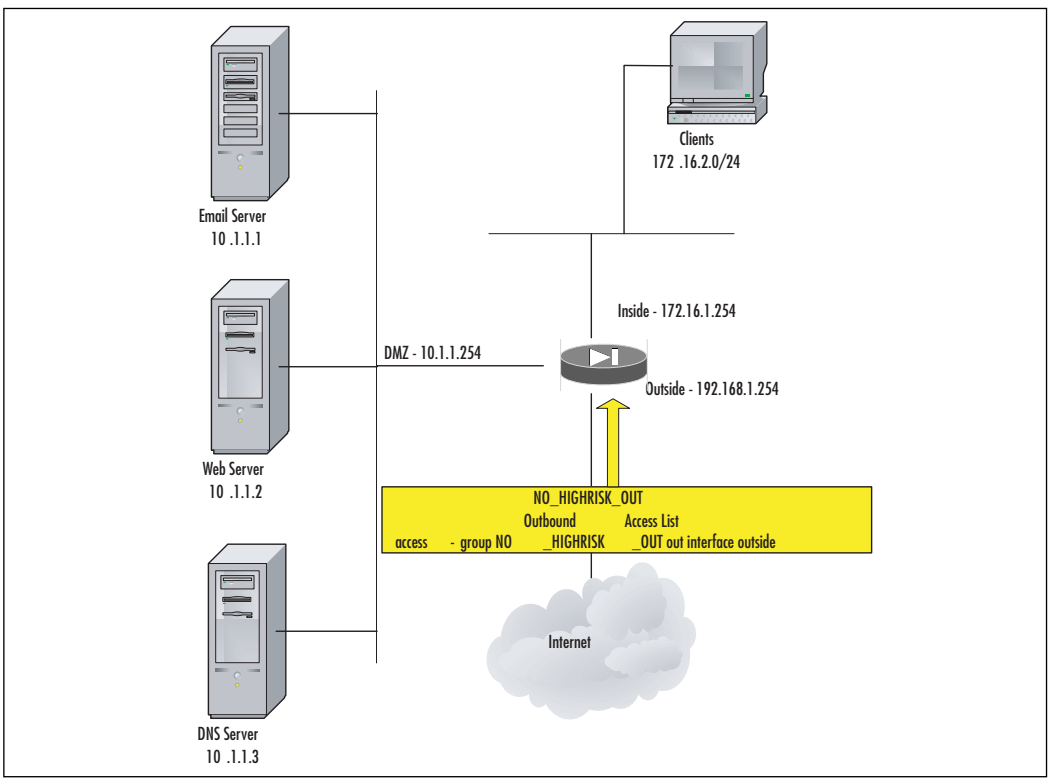
In the previous Secure Corporation example, we created and applied *inbound* access lists to the internal and DMZ interfaces to block this high-risk traffic from *entering* the PIX. Alternatively, we could create and apply a single *outbound* access list and apply it to the outside interface of the PIX to prevent the high-risk traffic from *exiting* the PIX. This solution, shown in Figure 3.6, is more scalable because the access list is created and applied once regardless of the number of interfaces. In previous versions, we would have had to include the relevant access list entries in a separate ACL applied to each interface.

For Secure Corporation, the Outbound ACL configuration follows:

```
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq 135
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq 135
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq netbios-ns
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq netbios-dgm
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq netbios-ssn
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq 139
```

```
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq 445
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq 445
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq tftp
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq bootpc
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq bootps
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq snmp
PIX1(config)# access-list NO_HIGHRISK_OUT deny udp any any eq snmptrap
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq sqlnet
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq 1214
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq 3408
PIX1(config)# access-list NO_HIGHRISK_OUT deny tcp any any eq 3531
PIX1(config)# access-list NO_HIGHRISK_OUT permit any any
PIX1(config)# access-group NO_HIGHRISK_OUT out interface outside
PIX1(config)# exit
```

Figure 3.6 Outbound ACL



Time-Based ACLs (New)

Version 7.0 introduces support for time-based ACLs, where individual access list entries can be configured to be active and enforced during a specified time period. This new capability has been implemented via a new command (*time-range*) and the extension of the existing *access-list* command with a new keyword (*time-range*). To implement time-based restrictions for an access list entry:

1. Define a time range via the new **time-range** command.
2. Create or modify an access list entry to use that time range via the **time-range** keyword in the **access-list** command.

The format of the **time-range** command is:

```
time-range name
```

The *name* parameter assigns a name to the time range you are defining. Once you enter this command, you enter time range configuration mode. Within this mode, you use the *absolute*, *periodic*, and *default* commands to define the time range parameters. The *absolute* command defines an absolute time when a time range is in effect. Its format is:

```
absolute [end time date] [start time date]
```

- The meaning of the *start* and *end* keywords is obvious.
- The format of the *time* parameters is *HH:MM* (e.g., 20:00 for 8 P.M.), and the format of the *date* parameters is *day month year* (e.g., 1 January 2006).

The *periodic* command defines a periodic time when the time range is in effect. Its format is:

```
periodic days-of-the-week time to [days-of-the-week] time
```

The parameters and keywords of the *periodic* command are identified and described here.

The first occurrence of the *days-of-the-week* parameter specifies the starting day or day of the week for the time range. The potential values for *days-of-the-week* are any single day or combinations of days, including Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. In addition, the following values are also valid:

- Daily
- Weekends
- Weekdays

The second occurrence specifies the ending day or day of the week for the time range. The second occurrence is optional and can be omitted if the ending days are the same as the starting days.

The first occurrence of the *time* parameter specifies the starting time, while the second occurrence specifies the ending time, and is *not* optional. The format of the *time* parameters

is *HH:MM* (e.g., 20:00 for 8 P.M.). Multiple *periodic* commands are permitted per *time-range* command. In addition, if a *time-range* command has both *absolute* and *periodic* values specified, the *periodic* commands are evaluated only after the *absolute start* time is reached, and are not further evaluated after the *absolute end* time is reached.

The *default* command restores the default configuration settings to the *time-range* command *absolute* and *periodic* keywords.

NOTE

Obviously, the time range feature relies on the accuracy of the PIX clock. Best practice would include the synchronization of the PIX clock with an NTP server.

Now that a time range has been defined using the *time-range* command, you must use it to specify an active time period for an access list entry via the *access-list* command. The general format of this command with respect to time ranges is:

```
access-list id [line line-number] [extended] {deny | permit} {tcp | udp}
{host sip | sip mask | any} [operator port] {host dip | dip mask | any}
[operator port] time-range time_range_name
```

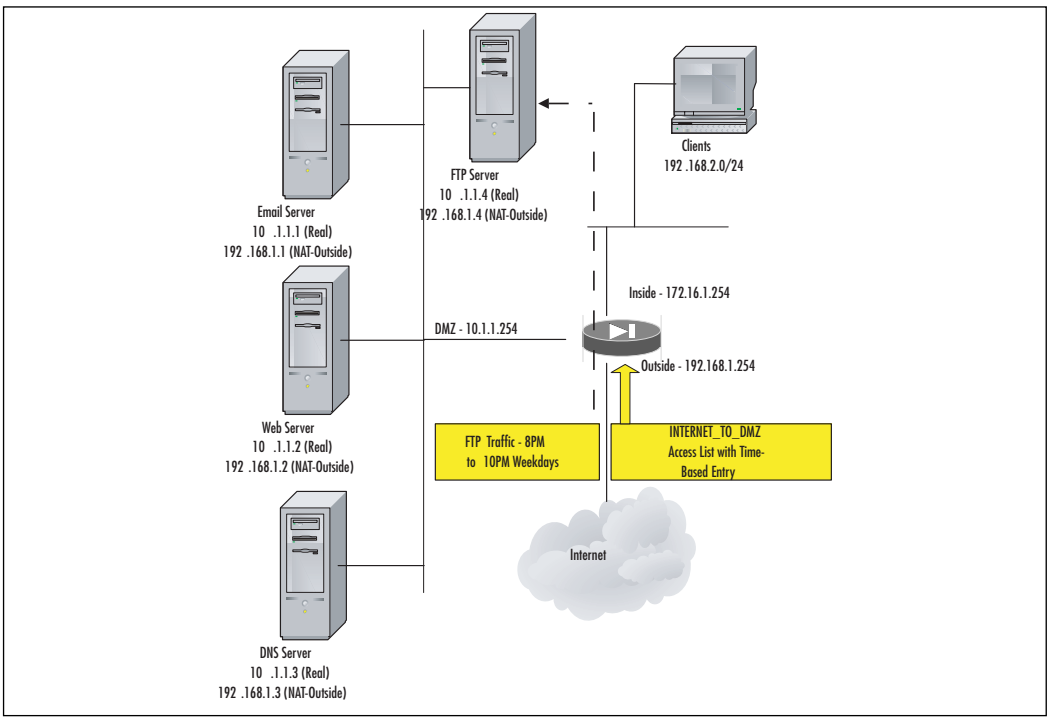
The use of access lists is discussed earlier in this chapter. To make an access-list statement active for a particular time range, simply include the *time-range* keyword and the *time_range_name*, which is the name of a time range previously defined using the *time-range* command.

For example, suppose Secure Corporation has a business requirement to exchange data with a partner via FTP. The company has implemented an FTP server in its DMZ to provide a staging point for the exchange of files, as shown in Figure 3.7. The exchange of data via FTP occurs nightly at a specified time. Because Secure Corporation does not want the DMZ FTP server exposed unnecessarily when it is not being used, it has chosen to implement a time-based ACL to allow FTP traffic to/from the server only when necessary.

The existing INTERNET_TO_DMZ access list has been extended via the following commands:

```
PIX1(config)# static (dmz, outside) 192.168.1.4 10.1.1.4 netmask
255.255.255.255 0 0
PIX1(config)# time-range PARTNER_FTP_TIME
PIX1(config-time-range)# periodic weekdays 20:00 to 22:00
PIX1(config-time-range)# exit
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any host 192.168.1.4 eq
ftp time-range PARTNER_FTP_TIME
PIX1(config)# access-list INTERNET_TO_DMZ permit tcp any host 192.168.1.4 eq
ftp-data time-range PARTNER_FTP_TIME
PIX1(config)# access-group INTERNET_TO_DMZ in Outside
PIX1(config)# exit
```

Figure 3.7 Time-Based ACL



NAT Control (New)

Version 7.0 simplifies the deployment of the PIX by eliminating the requirement for address translation policies to be in place before allowing network traffic to flow from a host on an inside network to outside networks. This feature is provided via a new command, ***nat-control***. When enabled, *nat-control* preserves the previous requirement that translation rules be defined before traffic can traverse the PIX from an inside interface to an outside interface. When disabled via the *no nat-control* command, the translation rules are not required for the traffic to flow from an inside interface to an outside one.

For new configurations, NAT Control is automatically disabled via the *no nat-control* command. For upgraded configurations, NAT Control is automatically enabled via the *nat-control* command to preserve the functionality already defined in the configuration.

If you have NAT Control enabled, but do not want to translate specific addresses, you can "bypass" NAT through one of several mechanisms described in the next section.

Bypassing NAT

With NAT Control, you can bypass NAT using one of three mechanisms that all achieve compatibility with inspection engines discussed in Chapter 5:

- Identity NAT (*nat 0* command)
- NAT exemption (*nat 0 access-list* command)
- Static identity NAT (*static* command)

Each of these mechanisms is described in the following paragraphs.

Identity NAT

Identity NAT is configured with the *nat 0* command. Instead of using an associated *global* command to define the global address, the internal address is mapped to itself when translating. Use the *nat* command with an *id* of 0, and do not define an associated *global* command. An example command to configure the PIX to translate any address in the 10.1.1.0/24 network to itself is:

```
PIX1(config)# nat (inside) 0 10.1.1.0 255.255.255.0
nat 0 10.1.1.0 will be non-translated
```

When configured, identity NAT applies to all interfaces. You cannot choose to perform identity NAT when the IP addresses access one interface (e.g., DMZ interface) and perform normal translation when the IP addresses access another interface (e.g., outside interface). Moreover, for identity NAT, connections can only be initiated from the inside to the outside. Connections from the outside cannot be established even if the interface access list allows it.

NAT Exemption

NAT exemption is configured with the *nat 0 access-list* command, which bypasses NAT altogether. First, you must define an access list that identifies the traffic to be translated. Then, use the *nat* command with an *id* of 0 and the access list name to bypass the NAT process. Example commands to configure the PIX to bypass NAT for the 10.1.1.0/24, 10.1.2.0/24, and 10.1.3.0/24 networks using an access list would be as follows:

```
PIX1(config)# access-list inside_public permit ip 10.1.1.0 255.255.255.0 any
PIX1(config)# access-list inside_public permit ip 10.1.2.0 255.255.255.0 any
PIX1(config)# access-list inside_public permit ip 10.1.3.0 255.255.255.0 any
PIX1(config)# nat (inside) 0 access-list inside_public
PIX1(config)# exit
PIX1# clear xlate
```

Like identity NAT, NAT exemption applies to all interfaces when it is configured. You cannot choose to perform NAT exemption when the IP addresses access one interface (e.g., DMZ interface) and perform normal translation when the IP addresses access another interface (e.g., outside interface). However, NAT exemption allows you to specify both the real and destination addresses (similar to policy NAT), so you have greater control than with identity NAT. Unlike identity NAT, NAT exemption permits connections to be initiated from both the inside and outside interfaces.

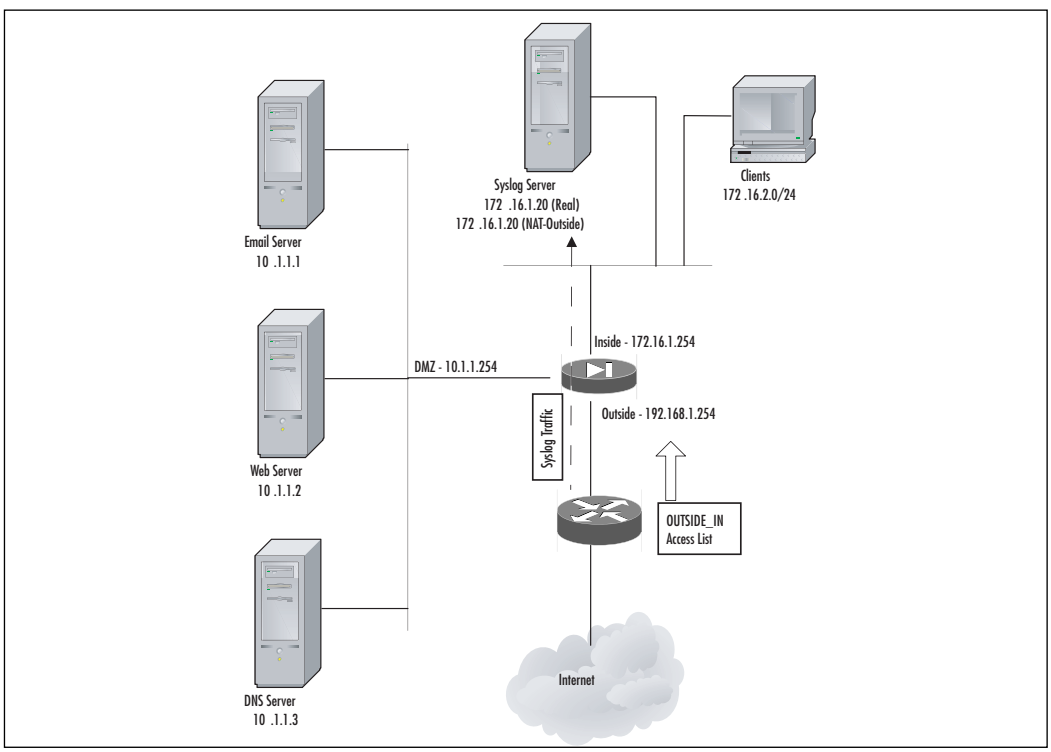
Static Identity NAT

Static identity NAT is configured using the *static* command. Unlike identity NAT and NAT exemption, static identity NAT does not apply to all interfaces when it is configured. You choose the interfaces for which you want the real address to appear. In addition, static identity NAT lets you use policy NAT, which specifies both the real and the destination addresses when determining translation.

For example, suppose you have an internal syslog server that you use to collect log information from all of your network perimeter devices, including your Internet router (see Figure 3.8). To allow syslog information from any device connected to a lower level security interface (e.g., Outside, DMZ) to the syslog server on the inside interface, you need to perform the steps for allowing inbound access:

1. Configure static address translation.
2. Define an access list permitting inbound syslog traffic from the Internet router to the internal syslog server.

Figure 3.8 Static Identity NAT



So far, these tasks should be nothing new to you. However, when configuring static address translation for the internal syslog server, you may not want to use one of your public

IP addresses. You probably have a limited number of public IP addresses, and you probably want to reserve them for devices that need to be accessible to a broader group of people (e.g., customers, partners). Using static *identity* NAT, you could simply define that *mapped* IP address to be the same as the *real* IP address. The commands to accomplish this are shown in the following example:

```
PIX1(config)# static (inside, outside) 172.16.1.20 172.16.1.20 netmask
255.255.255.255 0 0
PIX1(config)# access-list OUTSIDE_IN permit udp host 192.168.1.253 host
172.16.1.20 eq syslog
PIX1(config)# access-group OUTSIDE_IN in interface outside
PIX1(config)# exit
```

NOTE

For the preceding example to work, the Internet router needs to have a route to the internal syslog server (172.16.1.20) so that it knows which interface to send it out. Depending on your configuration, you may need to establish a static route to the outside PIX interface for the syslog server.

Policy NAT

Policy NAT lets you establish translation rules by specifying both the real address and the destination address. This allows you to translate addresses differently based on the destination of the packet. Policy NAT also allows you to translate addresses differently based on the source or destination ports.

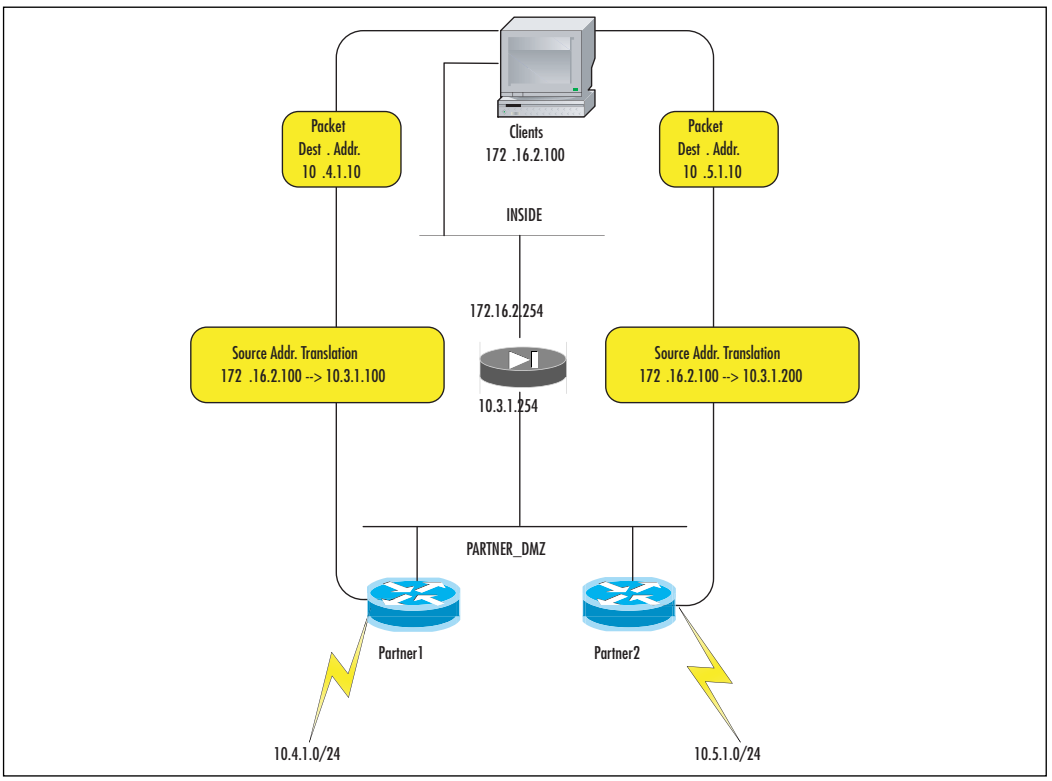
For example, suppose you have established a DMZ that is used strictly for partner connections. You have several private lines to partners that terminate on different routers within the DMZ, as shown in Figure 3.9. You have client workstations on your internal network that need to use both partner connections; however, your partners have placed requirements on you that dictate what source address you must use when accessing their resources. In other words, the same client workstation may be required to have two different source addresses when it communicates to your two partners.

You can accomplish this with policy NAT, which allows you to specify translations based on both source *and destination* addresses and ports. In the preceding example, you would accomplish this via the following commands:

```
PIX1(config)# access-list PARTNER1 permit ip 172.16.2.0 255.255.255.0
10.4.1.0 255.255.255.0
PIX1(config)# access-list PARTNER2 permit ip 172.16.2.0 255.255.255.0
10.5.1.0 255.255.255.0
PIX1(config)# nat (inside) 1 access-list PARTNER1
PIX1(config)# global (partner_dmz) 1 10.3.1.100 255.255.255.255
```

```
PIX1(config)# nat (inside) 2 access-list PARTNER2
PIX1(config)# global (partner_dmz) 2 10.3.1.200 255.255.255.255
PIX1(config)# exit
```

Figure 3.9 Policy NAT



NOTE

Except for NAT exemption, all forms of NAT support policy NAT. NAT exemption allows you to specify source and destination *addresses*, but not source and destination *ports*.

Object Grouping

Introduced in version 6.2, *object grouping* makes complex access lists much simpler to configure. Before the object-grouping feature was available, each unique network, node, service, and protocol combination that needed to be defined in an access list had to be configured

with a separate *access-list* statement. However, in most organizational security policies, groups of entries have similar access rights. Object groups allow groups of network addresses, services, protocols, and ICMP types to be defined, thereby reducing the number of access list entries needed.

For example, say that an organization wants to deny inside users access to a number of external FTP servers because they contain illegal software and viruses. Without object groups, an access list entry has to be defined for each individual FTP server. However, using object groups, we can define a network object group containing a list of hosts that contains all the IP addresses of the banned FTP servers. IP addresses can easily be added and removed from this group at will. Now, only one access list entry has to be created denying access to the object group from the inside. The access list does not need to be modified if entries are added or removed from the object group. As you can see, object groups allow for simplification of access list configuration and maintenance.

Configuring and Using Object Groups

There are four types of object groups: *icmp-type, protocol, network,* and *service*. Each object group type corresponds to a field in the *access-list* or *conduit* command. Once an object group has been created, a subconfiguration mode is entered so the group can be populated. Each object group type has different subconfiguration options, so we will look at each separately. Once an object group has been configured, it can be used in an *access-list* or *conduit* command.

ICMP-Type Object Groups

An *ICMP-type object group* is a group of ICMP-type numerical or literal values. ICMP-type object groups can be used in place of the *icmp-type* parameter in an access list or conduit. To create an ICMP-type object group, the syntax is:

```
object-group icmp-type <grp_id>
```

Once an object group has been defined, the subconfiguration mode enables the object group to be populated. At this stage, an optional description can be specified using the *description* subcommand. To populate the ICMP-type object group, the syntax is:

```
icmp-object <icmp_type>
```

For example, the following object group defines ICMP-type values that will be used later with an access list or conduit:

```
PIX1(config)# object-group icmp-type icmp-grp
PIX1(config-icmp-type)# description ICMP Type allowed into the PIX
PIX1(config-icmp-type)# icmp-object echo-reply
PIX1(config-icmp-type)# icmp-object unreachable
PIX1(config-icmp-type)# exit
PIX1(config)# exit
```

Network Object Groups

A *network object group* is a group of host IP addresses or networks. Network object groups can be used in place of an *src_addr* or *dst_addr* parameter in an access list or conduit statement. To create a network object group, the syntax is as follows:

```
object-group network <grp_id>
```

Network object groups have two subcommands for defining the group of hosts and networks. The syntax for defining a host entry in the object group is:

```
network-object host <host_addr | host_name>
```

The *host_addr* parameter is the IP address of the host being added to the object group. Alternatively, the *host_name* parameter specifies the hostname of a host defined using the *name* command.

The syntax for defining a network entry in the object group is:

```
network-object <net_addr> <netmask>
```

For example, the following object group defines host and network values to be used later with an access list or conduit:

```
PIX1(config)# object-group network net-grp
PIX1(config-network)# description List of Public HTTP Servers
PIX1(config-network)# network-object host 192.168.1.10
PIX1(config-network)# network-object host 172.16.10.1
PIX1(config-network)# network-object 172.16.2.0 255.255.255.0
PIX1(config-network)# exit
PIX1(config)# exit
```

Protocol Object Groups

A *protocol object group* is a group of protocol numbers or literal values. Protocol object groups can be used in place of the *protocol* parameter in an access list or conduit. To create a protocol object group, the syntax is as follows:

```
object-group protocol <grp_id>
```

Once an object group has been defined, the subconfiguration mode enables the object group to be populated. To populate the protocol object group, the syntax is:

```
protocol-object <protocol>
```

The *protocol* parameter is a protocol number or literal value. For example, the following object group defines a group of protocols that will be used later with an access list or conduit to provide VPN access:

```
PIX1(config)# object-group protocol vpn-grp
PIX1(config-protocol)# description Protocols allowed for VPN Access
```

```
PIX1(config-protocol)# protocol-object ah
PIX1(config-protocol)# protocol-object esp
PIX1(config-protocol)# protocol-object gre
PIX1(config-protocol)# exit
PIX1(config)# exit
```

Service Object Groups

A *service object group* is a group of TCP and/or UDP port numbers or port number ranges. Service object groups can be used in place of the *port* parameter in an access list or a conduit. The syntax to create a service object group is as follows:

```
object-group service <grp_id> tcp|udp|tcp-udp
```

Since a service object group is a listing of ports and port ranges, the ports defined need to be configured as TCP, UDP, or both TCP and UDP. The *tcp, udp*, and *tcp-udp* keywords define the common IP protocol for all ports listed in the object group. The subconfiguration command syntax to populate the service object group with a single port is:

```
port-object eq <port>
```

The subconfiguration command syntax to populate the service object group with a range of ports is:

```
port-object range <begin-port> <end-port>
```

For example, the following object group defines a group of ports that all Web servers within in organization need to have opened on the firewall:

```
PIX1(config)# object-group service websrv-grp tcp
PIX1(config-service)# description Ports needed on public web servers
PIX1(config-service)# port-object eq 80
PIX1(config-service)# port-object eq 8080
PIX1(config-service)# port-object range 9000 9010
```

To verify that an object group was created and populated with the correct information, we can view the current object group configuration using the *show object-group* command:

```
PIX1# show object-group
object-group icmp-type icmp-grp
  description: ICMP Type allowed into the PIX
  icmp-object echo-reply
  icmp-object unreachable
object-group network net-grp
  description: List of Public HTTP Servers
  network-object host 192.168.1.10
  network-object host 172.16.10.1
```

```
  network-object 172.16.2.0 255.255.255.0
object-group protocol vpn-grp
  description: Protocols allowed for VPN Access
  protocol-object ah
  protocol-object gre
  protocol-object esp
object-group service websrv-grp tcp
  description: Ports needed on public web servers
  port-object eq www
  port-object eq 8080
  port-object range 9000 9010
```

If one of the object groups does not look correct or is not needed, it can be removed using the *no object-group <grp_id>* command.

Object groups can be used in place of their respective values in access lists or conduits, but they must be preceded by the *object-group* keyword. For example, to allow the ICMP type values defined in the *icmp-grp* object group to enter the PIX's outside interface, the *access-list* command is:

```
PIX1(config)# access-list icmp_in permit icmp any any object-group icmp-grp
```

To allow access to the Web servers defined in the *net-grp* on the ports defined in *websrv-grp,* the command is:

```
PIX1(config)# access-list outside_in permit tcp any object-group net-grp
object-group websrv-grp
```

One nice feature of object groups is the ability to nest object groups of the same type together. For example:

```
PIX1(config)# object-group network all-servers
PIX1(config-network)# group-object net-grp
PIX1(config-network)# network-object 172.16.3.0 255.255.255.
```

TurboACLs

TurboACLs were a new feature in version 6.2 that enabled a long or complex access list to be *compiled,* or indexed, to enable faster processing of traffic through the access list. This was accomplished via the *compiled* keyword in the *access-list* statement, and could be turned on at the global level or for individual access lists.

With PIX software version 7.0, there is no longer a need to compile access lists. The software now automatically optimizes access list processing. From an upgrade perspective, any existing ***access-list*** statements with a *compiled* keyword are ignored and no longer accepted. An error message is printed and the statement is not stored in the running configuration, as shown here:

```
PIX1(config)# access-list compiled
ERROR:%  Incomplete Command
PIX1(config)# access-list 888 compiled
WARNING:%  This command has been DEPRECATED.  The access-lists are always
maintained in optimized form
```

Conduit and Outbound

PIX version 7.0 does not support the *conduit* and *outbound* commands, which have been completely replaced by the *access-list* command. From an upgrade perspective, you must migrate any *conduit* and *outbound* commands in your configuration file to *access-list* commands prior to upgrading to PIX v7.0. If you do not, the PIX will output errors.

You can migrate these commands in a completely manual fashion prior to the upgrade, or you can use the PIX Outbound/Conduit Converter (OCC) tool, which is available to contracted users form the Cisco Web site (www.cisco.com/pcgi-bin/tablebuild.pl/pix). This is for registered customers only.

This tool facilitates the conversion of *conduit* and *outbound* commands to access control list configurations; however, because of the different nature of these access control methods, there may be some changes to the actual functionality and behavior. This tool must only be considered as an aid and a starting point. You must review and test all configurations converted by the OCC tool prior to deployment.

Case Study

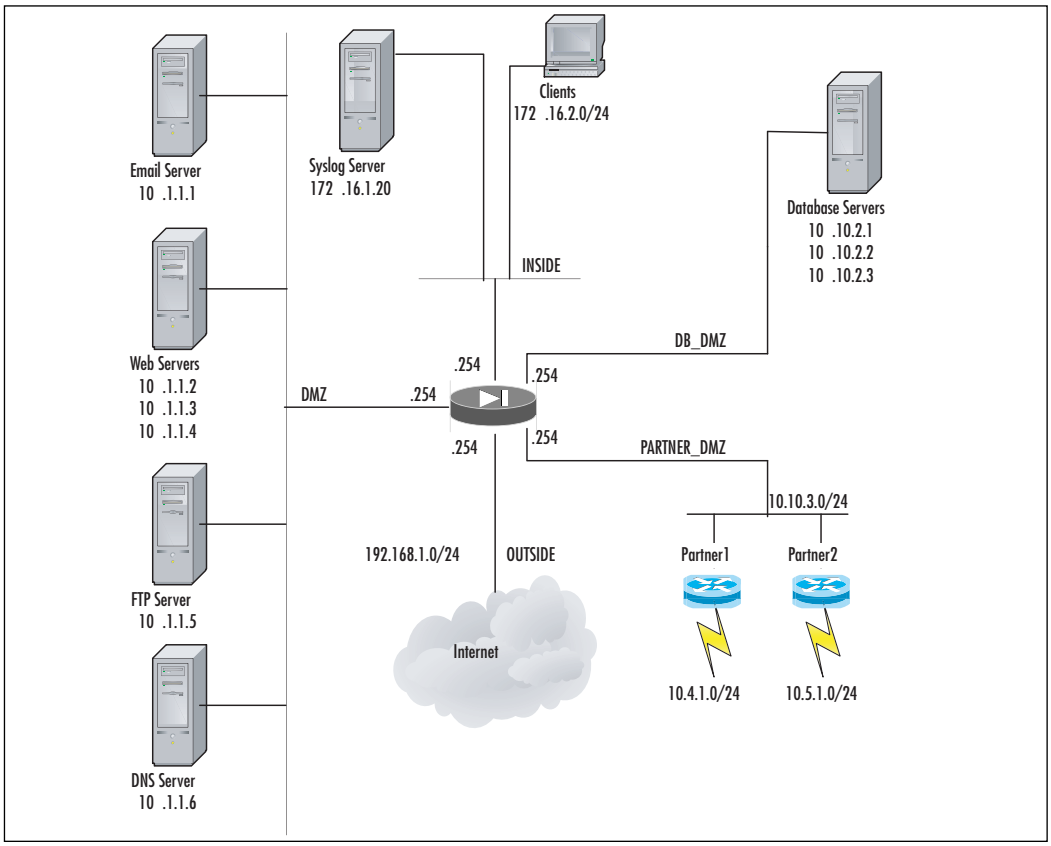
We've covered many important topics in this chapter. The following case study will put the concepts and features we learned into action.

Figure 3.10 shows the network layout of a new corporate site for Secure Corporation. The company has just bought the PIX and needs to configure it. It has already defined a security policy and has determined that the perimeter architecture it will implement requires four PIX interfaces.

The *inside* interface is the highest security interface. All corporate users and internal servers will be located behind this interface. Private addressing is used for the nodes located behind this interface. NAT Control will be used, and the PIX will use PAT to translate IP addresses when the nodes send traffic to the Internet. The PIX should not NAT any traffic from the nodes behind this interface when they access any other interface. There should be no direct access from the Internet to any server located behind this interface.

The *DMZ-DB* interface will have the second highest security level, and will be used to host database servers that enable the public Web servers to build dynamic HTML pages. No private or confidential information is stored on these database servers, which will use private addressing and are the only nodes located behind this interface. The database servers do not need access to the Internet, and no direct connections from the Internet should be allowed to the database servers. The database servers are using SQL*Net as the communication protocol with the Web servers; therefore, they need to be accessible from the Web servers in the DMZ. The database servers do not need direct access to any hosts on the inside interface.

Figure 3.10 A Complex Configuration of a Network



The *DMZ* interface will have the third highest security level. Publicly accessible services (Web, e-mail, and DNS) will be located behind this interface. In addition, an FTP server is used to transfer data files with partners. The file transfers occur each weeknight between 8 P.M. and 10 P.M., and the company wants to restrict access to the server to those time windows only. All the servers will use private addressing and require static translations. As these servers may be attacked, access to the Internet should only be allowed from the services that each server provides. Only direct access to the database servers from the Web servers using SQL*Net is permitted.

The *outside* interface will have the lowest security level. The company wants to only allow access to the identified services in the DMZ. The company also wants to make sure that it will not be the victim of a spoof attack, so it wants to filter out any traffic sourced with a private address. Since the inside network can ping, it is desirable to allow ICMP responses. In addition, the company has identified a list of high-risk traffic that it does not want to traverse from the inside, DMZ-DB, or DMZ networks to the Internet.

We will now discuss the commands to apply this security policy using NAT and access lists.

1. Begin by configuring the interfaces by naming them, assigning security levels, setting the speed and duplex, and assigning IP addressses:

```
PIX1(config)# interface ethernet0
PIX1(config-if)# nameif inside
PIX1(config-if)# security-level 100
PIX1(config-if)# speed 100
PIX1(config-if)# duplex full
PIX1(config-if)# ip address 172.16.1.254 255.255.240.0
PIX1(config-if)# no shutdown
PIX1(config-if)# exit
PIX1(config)# interface ethernet1
PIX1(config-if)# nameif outside
PIX1(config-if)# security-level 0
PIX1(config-if)# speed 100
PIX1(config-if)# duplex full
PIX1(config-if)# ip address 192.168.1.254 255.255.255.0
PIX1(config-if)# no shutdown
PIX1(config-if)# exit
PIX1(config)# interface ethernet2
PIX1(config-if)# nameif DMZ
PIX1(config-if)# security-level 40
PIX1(config-if)# speed 100
PIX1(config-if)# duplex full
PIX1(config-if)# ip address 10.10.1.254 255.255.255.0
PIX1(config-if)# no shutdown
PIX1(config-if)# exit
PIX1(config)# interface ethernet3
PIX1(config-if)# nameif DB_DMZ
PIX1(config-if)# security-level 80
PIX1(config-if)# speed 100
PIX1(config-if)# duplex full
PIX1(config-if)# ip address 10.1.2.254 255.255.255.0
PIX1(config-if)# no shutdown
PIX1(config-if)# exit
PIX1(config)# interface ethernet4
PIX1(config-if)# nameif PARTER_DMZ
PIX1(config-if)# security-level 60
PIX1(config-if)# speed 100
PIX1(config-if)# duplex full
PIX1(config-if)# ip address 10.1.3.254 255.255.255.0
PIX1(config-if)# no shutdown
PIX1(config-if)# exit
```

2. Assign a default route to the PIX:

```
PIX1(config)# route outside 0.0.0.0 0.0.0.0 192.168.1.1
```

3. Create an access list to be used later to bypass NAT:

```
PIX1(config)# access-list nonatinside permit ip 172.16.2.0
255.255.255.0 10.1.1.0 255.255.255.0
PIX1(config)# access-list nonatinside permit ip 172.16.2.0
255.255.255.0 10.1.2.0 255.255.255.0
PIX1(config)# access-list nonatdmzdb permit ip 10.10.2.0
255.255.255.0 10.1.1.0 255.255.255.0
```

4. Create a global pool using PAT for the inside network:

```
PIX1(config)# global (outside) 1 192.168.1.254
```

5. Bypass NAT where needed:

```
PIX1(config)# nat (inside) 0 access-list nonatinside
PIX1(config)# nat (DB_DMZ) 0 access-list nonatdmzdb
```

6. Enable NAT on the inside interface and have it mapped to the global *id*:

```
PIX1(config)# nat (inside) 1 0 0
```

7. Create static translations for access from the lower level security interfaces:

```
PIX1(config)# static (DMZ, outside) 192.168.1.11 10.1.1.1 netmask
255.255.255.255 0 0
PIX1(config)# static (DMZ, outside) 192.168.1.12 10.1.1.2 netmask
255.255.255.255 0 0
PIX1(config)# static (DMZ, outside) 192.168.1.13 10.1.1.3 netmask
255.255.255.255 0 0
PIX1(config)# static (DMZ, outside) 192.168.1.14 10.1.1.4 netmask
255.255.255.255 0 0
PIX1(config)# static (DMZ, outside) 192.168.1.15 10.1.1.5 netmask
255.255.255.255 0 0
PIX1(config)# static (dmz, outside) 192.168.1.16 10.1.1.6 netmask
255.255.255.255 0 0
PIX1(config)# static (DB_DMZ, DMZ) 10.1.1.11 10.1.2.1 netmask
255.255.255.255 0 0
PIX1(config)# static (DB_DMZ, DMZ) 10.1.1.12 10.1.2.2 netmask
255.255.255.255 0 0
PIX1(config)# static (DB_DMZ, DMZ) 10.1.1.13 10.1.2.3 netmask
255.255.255.255 0 0
PIX1(config)# static (inside, outside) 172.16.1.20 172.16.1.20
netmask 255.255.255.255 0 0
```

8. Configure names for the public addresses of the DMZ servers:

```
PIX1(config)# name 192.168.1.11 email
PIX1(config)# name 192.168.1.12 web1
PIX1(config)# name 192.168.1.13 web2
PIX1(config)# name 192.168.1.14 web3
PIX1(config)# name 192.168.1.15 ftp
PIX1(config)# name 192.168.1.16 dns
```

9. Configure object groups:

```
PIX1(config)# object-group network dbhosts
PIX1(config-network)# network-object host 10.1.2.1
PIX1(config-network)# network-object host 10.1.2.2
PIX1(config-network)# network-object host 10.1.2.3
PIX1(config-network)# exit
PIX1(config)# object-group network dmzhosts
PIX1(config-network)# network-object host 10.1.1.1
PIX1(config-network)# network-object host 10.1.1.2
PIX1(config-network)# network-object host 10.1.1.3
PIX1(config-network)# network-object host 10.1.1.4
PIX1(config-network)# network-object host 10.1.1.5
PIX1(config-network)# network-object host 10.1.1.6
PIX1(config-network)# exit
PIX1(config)# object-group network webhosts
PIX1(config-network)# network-object host 10.1.1.2
PIX1(config-network)# network-object host 10.1.1.3
PIX1(config-network)# network-object host 10.1.1.4
PIX1(config-network)# exit
PIX1(config)# object-group icmp-type icmp-outside-in
PIX1(config-icmp-type)# icmp-object echo-reply
PIX1(config-icmp-type)# icmp-object time-exceed
PIX1(config-icmp-type)# icmp-object unreachable
PIX1(config-icmp-type)# exit
PIX1(config)# object-group protocol tcphighrisk tcp
PIX1(config-protocol)# description highrisk tcp services
PIX1(config-protocol)# port-object eq 135
PIX1(config-protocol)# port-object eq 137
PIX1(config-protocol)# port-object eq 138
PIX1(config-protocol)# port-object eq 139
PIX1(config-protocol)# port-object eq 139
PIX1(config-protocol)# port-object eq 445
PIX1(config-protocol)# port-object eq 1521
PIX1(config-protocol)# port-object eq 1214
PIX1(config-protocol)# port-object eq 3408
```

```
PIX1(config-protocol)# port-object eq 3531
PIX1(config-protocol)# exit
PIX1(config)# object-group protocol udphighrisk udp
PIX1(config-protocol)# description highrisk udp services
PIX1(config-protocol)# port-object eq 67
PIX1(config-protocol)# port-object eq 68
PIX1(config-protocol)# port-object eq 69
PIX1(config-protocol)# port-object eq 135
PIX1(config-protocol)# port-object eq 137
PIX1(config-protocol)# port-object eq 138
PIX1(config-protocol)# port-object eq 139
PIX1(config-protocol)# port-object eq 139
PIX1(config-protocol)# port-object eq 445
PIX1(config-protocol)# port-object eq 161
PIX1(config-protocol)# port-object eq 462
PIX1(config-protocol)# exit
PIX1(config)# object-group network bogons
PIX1(config-network)# network-object 0.0.0.0 255.0.0.0.0
PIX1(config-network)# network-object 10.0.0.0 255.0.0.0.0
PIX1(config-network)# network-object 127.0.0.0 255.0.0.0.0
PIX1(config-network)# network-object 172.16.0.0 255.240.0.0.0
PIX1(config-network)# network-object 192.168.0.0 255.255.0.0.0
PIX1(config-network)# network-object 224.0.0.0 224.0.0.0.0
PIX1(config-network)# exit
```

10. Configure a time range for partner FTP access:

```
PIX1(config)# time-range PARTNER_FTP_TIME
PIX1(config-time-range)# periodic weekdays 20:00 to 22:00
PIX1(config-time-range)# exit
```

11. Configure access lists for each interface:

```
PIX1(config)# access-list dmzdb_in permit icmp 10.10.2.0
255.255.255.0 172.16.0.0 255.255.240.0
PIX1(config)# access-list dmzdb_in deny ip any any
PIX1(config)# access-list dmz_in permit tcp host 10.1.1.1 any eq
smtp
PIX1(config)# access-list dmz_in permit tcp host 10.1.1.6 any eq
domain
PIX1(config)# access-list dmz_in permit udp host 10.1.1.6 any eq
domain
PIX1(config)# access-list dmz_in permit tcp object-group dmzhosts
any eq http
```

```
PIX1(config)# access-list dmz_in permit tcp object-group webhosts
object-group dbhosts eq sqlnet
PIX1(config)# access-list dmz_in permit icmp object-group dmzhosts
172.16.0.0 255.255.240.0
PIX1(config)# access-list outside_in deny ip object-group bogons any
PIX1(config)# access-list outside_in permit tcp any object-group
webhosts eq http
PIX1(config)# access-list outside_in permit tcp host mail eq smtp
PIX1(config)# access-list outside_in permit tcp host dns eq domain
PIX1(config)# access-list outside_in permit udp host dns eq domain
PIX1(config)# access-list outside_in permit udp 192.168.1.253
172.16.1.20 eq syslog
PIX1(config)# access-list outside_in permit tcp any host ftp eq ftp
time-range PARTNER_FTP_TIME
PIX1(config)# access-list outside_in permit tcp any host ftp eq ftp-
data time-range PARTNER_FTP_TIME
PIX1(config)# access-list outside_in permit icmp any 192.168.1.0
255.255.255.0 object-group icmp-outside-in
PIX1(config)# access-list outside_in dny icmp any 192.168.1.0
255.255.255.0
PIX1(config)# access-list outside_in deny ip ip
PIX1(config)# access-list outside_out deny tcp any any eq object-
group tcphighrisk
PIX1(config)# access-list outside_out deny udp any any eq object-
group udphigh risk
PIX1(config)# access-list outside_out permit ip any any
```

12. Apply the access lists to the appropriate interfaces:

```
PIX1(config)# access-group outside_in in interface outside
PIX1(config)# access-group outside_out out interface outside
PIX1(config)# access-group dmz_in in interface dmz
PIX1(config)# access-group dmzdb_in in interface DB_DMZ
```

13. Configure Policy NAT for partner DMZ connections:

```
PIX1(config)# access-list PARTNER1 permit ip 172.16.2.0
255.255.255.0 10.4.1.0 255.255.255.0
PIX1(config)# access-list PARTNER2 permit ip 172.16.2.0
255.255.255.0 10.5.1.0 255.255.255.0
PIX1(config)# nat (inside) 11 access-list PARTNER1
PIX1(config)# global (partner_dmz) 11 10.3.1.100 255.255.255.255
PIX1(config)# nat (inside) 12 access-list PARTNER2
PIX1(config)# global (partner_dmz) 12 10.3.1.200 255.255.255.255
```

Summary

Configuring the PIX to pass inbound or outbound traffic requires multiple steps. Basic connectivity allows users on a higher security-level interface of the PIX to transmit traffic to a lower security-level interface using NAT or PAT. This is accomplished using the *nat* command in conjunction with a *global* command. Because the PIX allows higher security-level interfaces to transmit traffic to lower security-level interfaces, and because the PIX is stateful, users on the inside of the PIX should be able to run almost any application without extra configuration on the PIX.

Controlling outbound traffic has become a critical part of a comprehensive security policy. With PIX version 7.0, this can be accomplished only using the *access-list* and *access-group* commands. The *outbound* command is no longer supported. In addition, if you keep traffic that you consider high risk from leaving your network, including the inside and DMZ interfaces, you can use the Outbound ACL feature. This feature is new in PIX version 7.0 and allows you to apply an access list either inbound or *outbound to the interface*. To prevent high-risk traffic from leaving your network, you can simply create and apply a single Outbound ACL to the outside interface instead of creating and applying multiple inbound ACLs to each of your internal interfaces.

Once outbound access is secure, moving on to allowing inbound access is relatively easy. By default, all inbound access (connections from a lower security-level interface to a higher security-level interface) is denied. With PIX version 7.0, only access lists can be used to allow inbound traffic. Conduits are no longer supported. The fundamentals of the *access-list* command are no different between controlling inbound or outbound traffic. For inbound traffic, configuring a static translation (using the *static* command) is required for each publicly accessible server in addition to *access-list* or *conduit*.

With PIX version 7.0, there are several new access list features, including time-based ACLs and the ability to enable/disable individual access control list entries. The time-based ACL feature allows you to specify a time period during which an access control list entry is active. This could be useful for permitting access to DMZ resources to partners for scheduled jobs only during a specified time period. The ability to enable/disable individual access control list entries should ease troubleshooting for complex ACLs.

Version 7.0 also simplifies the deployment of the PIX by eliminating the requirement for address translation policies to be in place before allowing network traffic to flow from a host on an inside network to outside networks. This feature is provided via a new command, *nat-control*. When enabled, *nat-control* preserves the previous requirement that translation rules be defined before traffic can traverse the PIX from an inside interface to an outside interface. When disabled via the *no nat-control* command, the translation rules are not required for the traffic to flow from an inside interface to an outside one. If NAT Control is desired or required by your security policy, you can selectively bypass NAT using one of three mechanisms: identity NAT (*nat 0* command), NAT exemption (*nat 0 access-list* command), and static identity NAT (*static* command).

Policy NAT lets you establish translation rules by specifying both the real address and the destination address. This allows you to translate addresses differently based on the destination of the packet. Policy NAT also allows you to translate addresses differently based on the source or destination ports.

Object grouping makes complex access lists much simpler to configure. Without object grouping, each unique network, node, service, and protocol combination that needs to be defined in an access list has to be configured with a separate access-list statement. However, in most organizational security policies, groups of entries have similar access rights. Object groups allow groups of network addresses, services, protocols, and ICMP types to be defined, thereby reducing the number of access list entries needed.

Solutions Fast Track

NAT Control

- ☑ Version 7.0 simplifies the deployment of the PIX by eliminating the requirement for address translation policies to be in place before allowing network traffic to flow from a host on an inside network to outside networks.
- ☑ This feature is provided via a new command, *nat-control*. When enabled, *nat-control* preserves the previous requirement that translation rules be defined before traffic can traverse the PIX from an inside interface to an outside interface. When disabled via the *no nat-control* command, the translation rules are not required for the traffic to flow from an inside interface to an outside one.
- ☑ For brand new PIX installations, NAT control is automatically disabled via the *no nat-control* command. For an upgrade of an existing configuration, NAT control is automatically enabled via the *nat-control* command to preserve the functionality already defined in the configuration.
- ☑ If NAT Control is desired or required by your security policy, you can selectively bypass NAT using one of three mechanisms: identity NAT (*nat 0* command), NAT exemption (*nat 0 access-list* command), and static identity NAT (*static* command).

The Bare Minimum: Outbound Traffic

- ☑ By default, if address translation is configured, the PIX firewall allows all connections from a higher security-level interface to a lower security-level interface.
- ☑ A well-defined security policy usually does not allow all outbound traffic. Define and control which applications you allow.
- ☑ With PIX version 7.0, there is only one method for controlling outbound traffic: access lists. The *outbound* and *apply* commands are no longer supported.
- ☑ PIX version 7.0 also introduces a feature called Outbound ACLs. In previous PIX OS versions, access lists could be applied to only packets that were *inbound* to the *interface*. With the PIX v7.0 Outbound ACL feature, access lists can be applied to either *inbound to* or *outbound from* the interface. However, you can only apply one ACL per direction on each interface.

- ☑ PIX version 7.0 introduces a new feature called "time-based ACLs," which allow you to specify a time period during which an access control list entry is active. This could be useful for permitting access to DMZ resources to partners for scheduled jobs only during a specified time period.
- ☑ PIX version 7.0 introduces a new feature that allows you to enable/disable individual access control list entries. The ability to enable/disable individual access control list entries should ease troubleshooting for complex ACLs.
- ☑ With PIX version 7.0, TurboACLs are no longer supported. TurboACLs was a new feature in version 6.2 that enabled a long or complex access list to be *compiled,* or indexed, to enable faster processing of traffic through the access list. There is no longer a need to compile access lists because the software now automatically optimizes access list processing.

Opening Your Network: Allowing Inbound Traffic

- ☑ By default, connections from a lower security-level interface to a higher security-level interface are denied.
- ☑ Port redirection is an excellent option for small businesses that do not have the money to buy a large amount of IP address ranges.
- ☑ The syntax for access lists is the same whether they are applied to inbound or outbound traffic.

Object Grouping

- ☑ Object groups allow simplification of access list configuration and management.
- ☑ There are four types of object groups: ICMP type, network, protocol, and service.

Frequently Asked Questions

Q: Could I use a *static* command with a *netmask* option instead of the *nat 0 access-list* command to configure public IP addresses inside the PIX?

A: Although this configuration will work, it opens the firewall to vulnerabilities if an access list is misconfigured. Use *nat 0 access-list* if you can.

Q: Why do I have to issue a *clear xlate* after I make changes?

A: The *xlate* table is maintained by the NAT process of the PIX, so if you make changes to that process, items can become stuck in the table, or items that should not be in the table might still remain. This can cause unpredictable results, and creates a security risk.

Q: Should I move all my servers into a DMZ?

A: DMZs are very helpful in containing security risks for publicly accessible servers. If a server is not accessible to the outside world, there is probably no good reason to move it into a DMZ. If you do not trust the inside users, that is another story.

Q: Why should I use private IP addresses inside my network if I have enough public address space?

A: Using private address space inside your network can provide many advantages to a corporation. The amount of address space provided allows for increased flexibility in the network design and allows for expansion. However, private addresses are not for everyone, and many universities and other institutions that have large amounts of IP address space use public addressing in their networks.