



Interested in learning more about security?

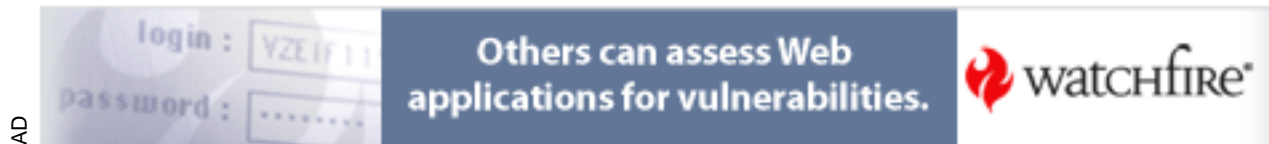
SANS Institute InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

The Installation and Configuration of a Cisco PIX Firewall with 3 Interfaces and a Stateful Failover

This paper is intended to guide the reader through the installation and configuration of a Cisco PIX firewall. The configuration consists of inside, outside, and a DMZ network. A fourth interface will be used to provide a high-availability stateful failover situation. In this paper I will be using a Cisco PIX Model 525 firewall running software version 6.2. I do not review licensing issues in this paper. My intended audience are those who know the basics of a firewall, and have general PC knowledge. I also assume that ...

Copyright SANS Institute
Author Retains Full Rights



Steve Textor
Version 1.4
April 29, 2002

The Installation and Configuration of a Cisco PIX Firewall with 3 Interfaces and a Stateful Failover Link

Introduction

This paper is intended to guide the reader through the installation and configuration of a Cisco PIX firewall. The configuration consists of inside, outside, and a DMZ network. A fourth interface will be used to provide a high-availability stateful failover situation. In this paper I will be using a Cisco PIX Model 525 firewall running software version 6.2. I do not review licensing issues in this paper. My intended audience are those who know the basics of a firewall, and have general PC knowledge. I also assume that the reader knows some basic Cisco configuration commands. They should know what the user mode and privilege modes are and well as the configuration mode. They also should know the concepts behind routing, filtering, TCP/IP, subnetting, ports, and terminology of networking in general.

Conventions

Command descriptions used within this document.

- **Boldface** indicates literal commands that you type in as shown
- *Italics* indicate arguments to the commands.

PIX – An Introduction

A Cisco PIX is a firewall device that runs a hardened, specially built operating system. The OS of the PIX was specifically designed for a highly protected, secure and redundant environment.

The PIX firewall protects the inside network from unauthorized access from an external source, such as the Internet. Most firewalls have one or more perimeter networks or demilitarized zones (DMZ). A DMZ is a perimeter network that is typically less restrictive than access into the internal network. Connections between the inside, outside,

and DMZ (perimeter) networks are controlled by the PIX, as all network traffic must pass through the PIX to get from network to network.

The PIX offers security for a wide range of network services, include the following:

- Network Address Translation (NAT)
- IPsec VPN
- DHCP client/server
- PPOE (Point-to-Point over Ethernet)
- Content and URL filtering
- Special handling of DNS, H.323, and RealAudio so that these popular protocols work in a secure manner across the PIX

The PIX protection is based on the Adaptive Security Algorithm (ASA). ASA is a stateful approach to firewall protection. Meaning:

- Every inbound packet is checked against the ASA and against the session flow table. Also inbound connections are implicitly denied unless allowed by the firewall rules, a.k.a access control list (ACL).
- ASA allows outbound connections without an ACL specifically allowing that connection.
- ASA also randomizes the TCP sequence numbers to best protect against a TCP sequence number attack.
- ASA checks the source IP address, source port, destination IP address, destination port, TCP sequence number and any additional TCP flags of each packet.
- With ASA, no packets can traverse the PIX without a TCP SYN packet establishing a connection or the connection already exist in the translation table.
- All ICMP packets are denied unless specifically allowed.
- All attempts to circumvent these rules are dropped and logged.

ASA or Stateful packet filtering works like this:

When a firewall receives a TCP SYN packet requesting a connection, the firewall reads its rules in sequential order from top to bottom. If the request matches a rule then a connection is established. When a connection is established (rule match) the PIX firewall creates a translation slot in the state table thus creating a session flow. If a packet comes across that is not a TCP SYN packet, the packet's information is checked against the state table to see if a session flow is already established. If a session has already been established the packet is allowed through and the rules are not checked. If a rule is not matched, the connection is denied and a RST (Reset) packet is sent back to the originator. As you can see, there is less sequential reading of the firewall rules when a connection table exists. Less "reading" means faster transfer of packets.

Another way of blocking packets is through stateless packet filtering. Stateless filtering involves the reading the source, and destination IP addresses of a TCP packet, then cross-referencing the rule-base to see if a connection should be established. This is done each time for every packet in a session. Packet filtering is not as fast and robust as stateful

packet filtering. One of the bigger advantages of stateful packet filtering over stateless is that stateful filtering can detect incorrectly sequenced numbered packets or inconsistent options in the IP protocol, which of course protects the network

Yet another firewall type is a proxy. When a connection is established between two computers, say a client and a server. That connection is actually established between only those two computers. A proxy breaks up this connection and establishes a new session on behalf on the client. The proxy will then check the entire packet, application data and all, for any malformed data then pass it on to the real end-client. All this checking of data in each packet tends to make the proxy solution slow.

PIX uses a type of proxy called a cut-through proxy. A cut-through proxy challenges a user initially at the application layer, like a traditional proxy. But once the user is authenticated a session flow is created and subsequent packets are checked against the state table as in the above explanation.

The Scenario

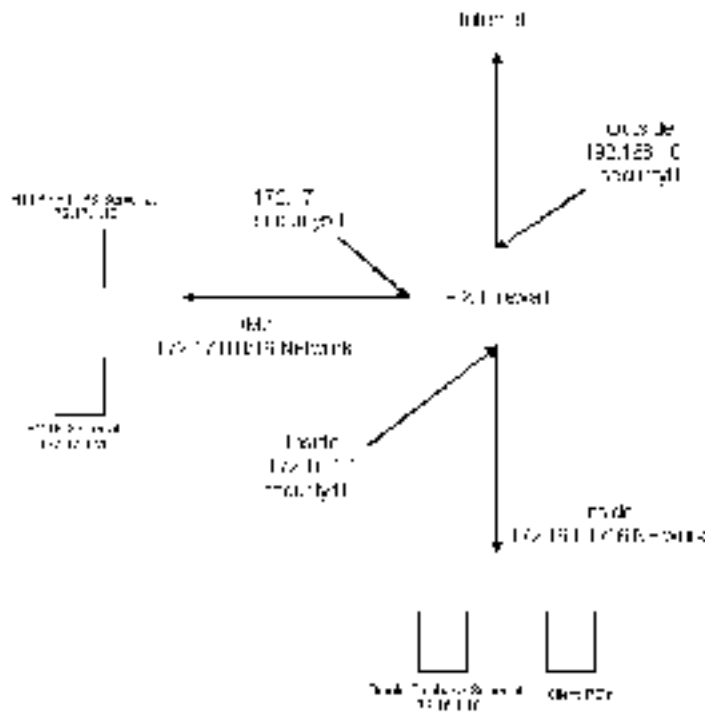
You are a medium size company with 1000 employees. You are asked to setup a firewall with a stateful failover configuration. You have only a DMZ network, one internal network and the external network is the Internet. The networks are all Ethernet 100baseT. Your users need Internet access for HTTP, HTTPS, and FTP. You have an Oracle database in your internal network that is queried by an application on your web server. Here is the pertinent external and internal IP addresses:

- Traffic for HTTP and HTTPS to external address 192.168.10.10 goes to the internal 172.17.1.10 IP address.
- Oracle database traffic (port 1521) needs to be passed between DMZ address 172.17.1.10 and internal address 172.16.1.10
- Internal users are allowed FTP and WWW to the outside. They also use Email.
- The SMTP server is at DMZ address 172.17.1.20 and its external address is 192.168.10.20.

We are going to have to use a little imagination here. In an effort to sanitize this document, I have changed all references to my real registered IP addresses to the private address class network of 192.168.10.0/24. So in the configuration, anytime you see the 192.168.10.0/24 network, substitute your real globally unique IP addresses.

[Figure 1](#) illustrates what your firewall setup looks like:

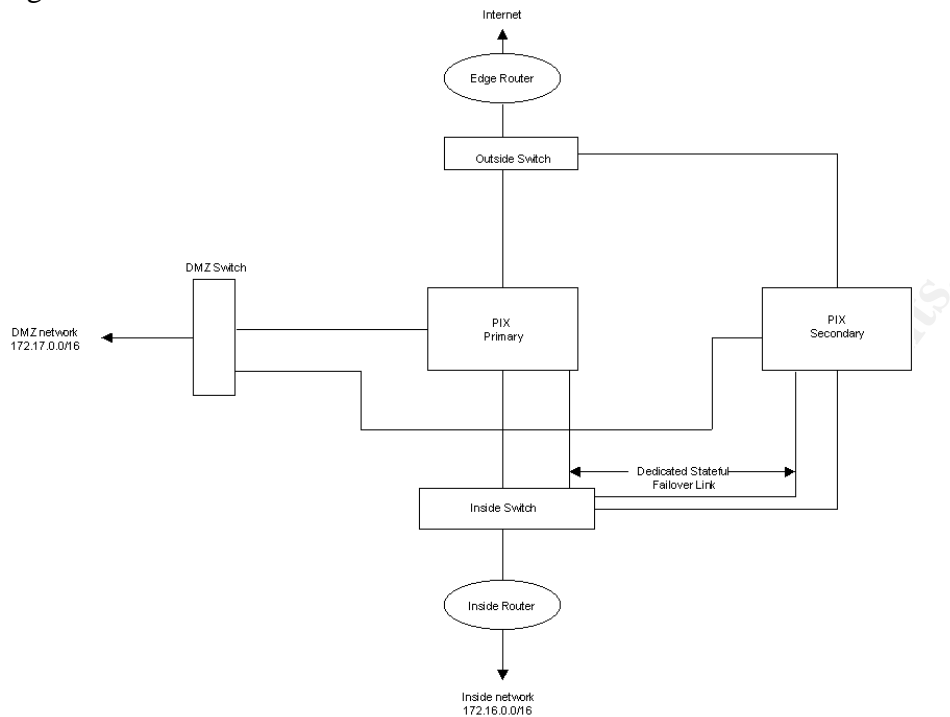
Figure 1



Getting Started

Since the company desires a stateful failover situation, the easiest thing to do is to first put together the hardware. We want to connect all of our cat5 Ethernet cables, the failover cable, and the failover link crossover cable. At this time we only want to power on the primary PIX. Do not turn on the secondary unit until told to do so. Just pick one of the PIX's to be primary, it doesn't matter since they are both the same. Next plug in all of your RJ-45 cat5 Ethernet cables into the interfaces on both PIX's. Then plug those cables into your switches (as depicted in [Figure 2](#)). You must use the same interfaces on both PIX's for the same purpose. In other words, you will be using ethernet0 for the outside so ethernet0 on both PIX's must be plugged into the "outside switch" and follow this procedure for all interfaces (again see [Figure 2](#)).

Figure 2

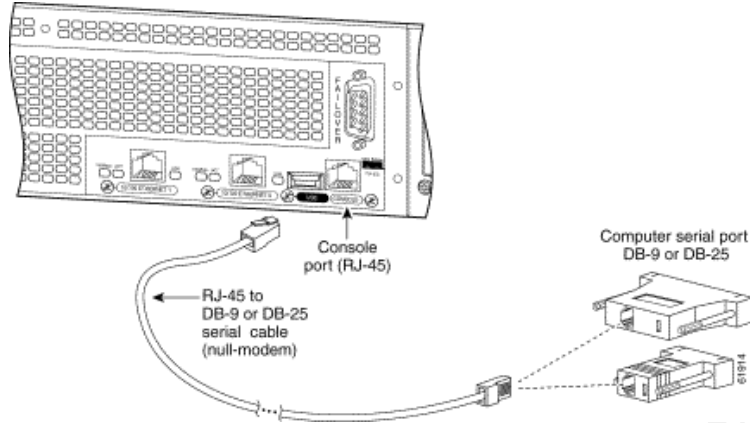


The interface's cables coming from each PIX and going to the same switch must be on the same VLAN and set to the same speed and duplex. Now we'll work on the stateful failover link. It will be a crossover cabled directly between the PIX's on the same interface. We will configure both PIX's from only one unit, the primary, so do not turn on the secondary PIX yet.

The first thing we have to do to the PIX is get to the command line so we can configure it. We would normally get to the command line via telnet, but you need to type commands to set the PIX up with an IP address and tell it what is allowed to telnet in. So we must initially connect via the console port.

Connecting to the PIX via the console port is and relatively easy task. You need a PC or laptop within cable length to the PIX.

Figure 3



Picture from Cisco Website at URL:

http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_61/pix_ig/525.htm#57873

Located within the accessory kit that comes with each PIX purchase you will find a serial cable assembly kit. The serial cable assembly consists of a null modem cable with RJ-45 connectors, and one DB-9 connector and a DB-25 connector. Assemble this and put the RJ-45 end into the console port on the back of the PIX and connect the other end into your serial port on your PC as depicted in [Figure 3](#).

Next you will need a terminal program like Microsoft Windows HyperTerminal. Open this program and configure it to connect directly to COM 1 (or whatever serial com port you connected the serial cable to). Set COM 1 to 9600 bits per second, 8 data bits, no parity, stop bits to 1, and hardware flow control. Connect to the PIX using this configuration. On a fresh PIX with its default configuration you should see the following prompt:

```
pixfirewall>
```

This prompt indicates by the chevron that you are in unprivileged mode. Type in the word **enable** then the enter key to go into privileged mode. When you type in **enable** you will be asked to supply a password. There is no “enable password” at the moment so just hit enter. This will provide the following prompt:

```
pixfirewall#
```

The pound sign (#) indicates that you are in privileged mode. If you type in **write term** it will display the configuration on the terminal. A default configuration looks much like this ([Figure 4](#)):

Figure 4

```
pixfirewall# wr t
Building configuration...
: Saved
:
PIX Version 6.2
nameif ethernet0 outside security0
nameif ethernet2
nameif ethernet3
nameif ethernet1 inside security100
enable password xxxxxxxxxxxxxxx encrypted
passwd xxxxxxxxxxxxxxx encrypted
hostname pixfirewall
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
names
no pager
logging on
no logging timestamp
no logging standby
no logging console
no logging monitor
no logging buffered
no logging trap
no logging history
logging facility 20
logging queue 512
interface ethernet0 auto shutdown
interface ethernet1 auto shutdown
mtu outside 1500
mtu inside 1500
ip audit info action alarm
ip audit attack action alarm
pdm logging informational 100
pdm history enable
arp timeout 14400
timeout xlate 0:05:00
```

```
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h323 0:05:00 sip
0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
http server enable
no snmp-server location
no snmp-server contact
snmp-server community public
no snmp-server enable traps
floodguard enable
telnet timeout 5
ssh timeout 5
terminal width 80
Cryptochecksum:6380fd60019789e34aac0d059a26f
: end
[OK]
pixfirewall#
```

This is a PIX default configuration, I will go over the commands to get the PIX up and filtering packets. If you are not sure that the configuration is at its default settings or you just want to make sure, then you can enter the **write erase** and then the **reload** command to clear the configuration.

First we need to configure some preliminary settings. We will set a user mode or telnet password, a privilege mode password, and a host name. At the PIX go into enable mode by typing **enable**. The enable and privilege mode are the same and I will use them interchangeably. You should now be at a pixfirewall# prompt. Now go into configuration mode by typing **config terminal** (or **config t** for short). You will run most of your commands to setup the PIX in this mode.

Next let's name our PIX and give it some passwords. While in **config t** mode type in:

```
pixfirewall(config)# hostname PIX1
PIX1(config)# enable password enable-password-here
PIX1(config)# passwd telnet-password-here
PIX1(config)# write mem
```

Notice how our prompt changed after we issued the **hostname** command. The **enable password** command sets a password to enter in enable or privilege mode. The **passwd** command sets a login or user mode password. The last command writes the configuration to the startup configuration, or more simply saves your work.

The next thing you want to do is name your interfaces. Looking at our config in [figure 4](#) we see that we have four name interface or **nameif** statements. Two of them are already

named; the inside and the outside and they already have a security level assigned to them. The outside interface takes the default values of ethernet0 and a security level of 0 (zero). The inside interface uses the default values of Ethernet1 and a security level of 100. The security levels work like this:

The outside security level is 0 and the inside security level is 100 by default. The levels can be changed, but I wouldn't recommend it. All other interfaces can take any number between 1 and 99. It is common and good practice to set other interfaces at values at least 5 or 10 numbers apart. To pass traffic from a higher security level to a lower security level, nat and global or static commands are needed. To go from a lower level to a higher level, static and access-list commands are needed. And finally, interfaces with the same security level cannot communicate with each other.

Since the inside and outside interfaces are already named and given security levels, all we need to do is to give ethernet2 and ethernet3 names and a security level. We will give ethernet2 the name of DMZ and a security level of 50. We will use our last interface, ethernet3, as our stateful failover link. The security level for the Failover interface doesn't matter as long as it is between 1 – 99 and no other interface is using the same level. I will go over Failover and Stateful Failover later. To set these interfaces use the nameif command. The name interface command takes on the syntax of **nameif hardware_id if_name security_level** . So type in:

```
PIX1(config)# nameif ethernet2 DMZ sec50  
PIX1(config)# nameif ethernet3 SFailover sec20
```

Now we want to use the interface command and set the speed, and duplex of our interfaces. Also, we use the interface command to turn the interfaces on, as they are currently in a “shutdown” state. We'll set them to a speed of 100 Mbps and full duplex. There are some rules that must be followed. Since we are going to setup a stateful failover, we need to have that interface (ethernet3) setup to be at least 100full or 1000full on both PIX's. Actually the rule states that our failover link needs to be equal to or greater than the fastest interface on our PIX starting with a speed of 100. So, if we have all gigabit interfaces, our failover link needs to be at least a gigabit interface. Let's set the interfaces up with these commands:

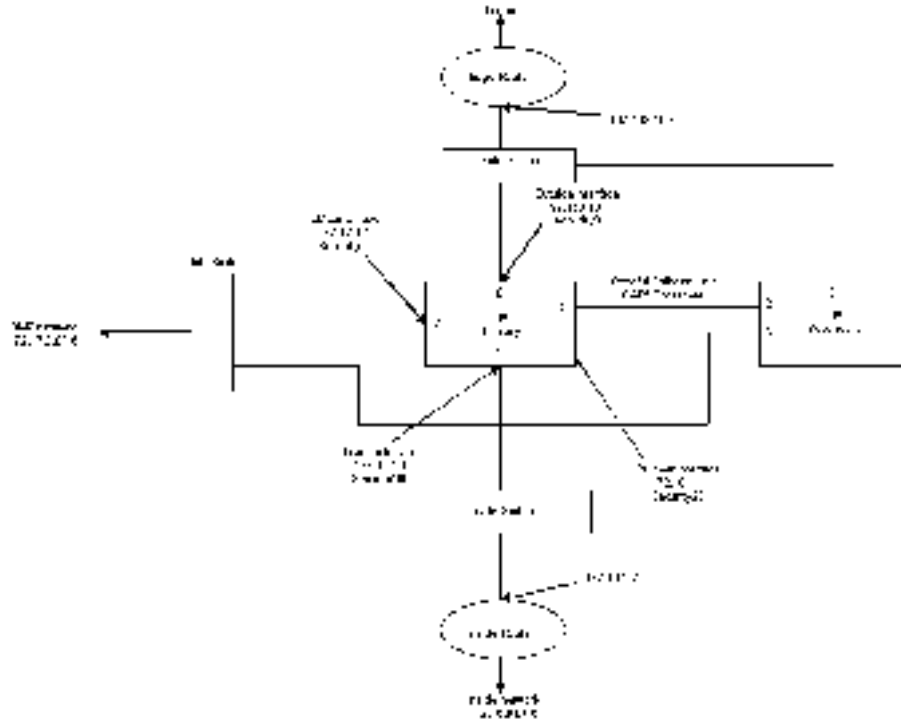
```
PIX1(config)# interface ethernet0 100full  
PIX1(config)# interface ethernet1 100full  
PIX1(config)# interface ethernet2 100full  
PIX1(config)# interface ethernet3 100full
```

In the above commands the 100full turns on the interface. There is no need to issue a no shutdown command as we do on an IOS router. We could have used the attribute of *auto* as well, but you can not use auto on the failover link. Note that the 100full is typed as one word. We don't have any extra interfaces on our PIX, but if we did it is good practice to name the interface (**nameif** command), make sure it is shutdown and give it the

127.0.0.1 IP address. This trick prevents passing of traffic if the interface is turned on accidentally. I would name the interface something like “unused”.

Now we need to give the interfaces some IP addresses. Let’s get a look at our physical network at [Figure 5](#).

Figure 5



Our DMZ is on the 172.17.0.0/16 network, our inside is the 172.16.0.0/16 network, and our outside is the Internet. Let’s first concentrate on the DMZ network. When a server on the 172.17.0.0/16 network needs to send data outside of its network (say the outside) then they will have to send it to its default gateway (a router). The PIX is a not a router by nature but it can be setup to route. So when any of the servers on the DMZ need to send data outside of their network they will send it to their default gateway. This will be the DMZ interface on the PIX. The PIX will then pick it up and know what to do with it, or it will know what to do with it when we configure routing on it. We’ll make this default gateway 172.17.1.1. This means that on the PIX the DMZ interface will have the IP address of 172.17.1.1 with a mask of 255.255.0.0. Let’s setup all the IP addresses for the interfaces. The syntax for the command is **ip address interface_name ip_address subnet_mask**. Type in:

```
PIX1(config)#ip address dmz 172.17.1.1 255.255.0.0
PIX1(config)#ip address inside 172.16.1.1 255.255.0.0
PIX1(config)#ip address outside 192.168.10.1 255.255.255.0
PIX1(config)#ip address sfailover 172.18.1.1 255.255.255.0
```

Let's recap with another look at our configuration. At the prompt type in **wr t**, to write the configuration to the terminal. You should see the following (Figure 6):

Figure 6

```
PIX1(config)# wr t
Building configuration...
: Saved
:
PIX Version 6.2
nameif ethernet0 outside security0
nameif ethernet2 DMZ security50
nameif ethernet3 SFailover security20
nameif ethernet1 inside security100
enable password xxxxxxxxxxxxxxx encrypted
passwd xxxxxxxxxxxxxxx encrypted
hostname PIX1
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
names
no pager
logging on
no logging timestamp
no logging standby
no logging console
no logging monitor
no logging buffered
no logging trap
no logging history
logging facility 20
logging queue 512
interface ethernet0 100full
interface ethernet1 100full
interface ethernet2 100full
interface ethernet3 100full
mtu outside 1500
mtu inside 1500
mtu dmz 1500
mtu sfailover 1500
```

```
ip address dmz 172.17.1.1 255.255.0.0
ip address inside 172.16.1.1 255.255.0.0
ip address outside 192.168.10.1 255.255.255.0
ip address sfailover 172.18.1.1 255.255.255.0
ip audit info action alarm
ip audit attack action alarm
pdm logging informational 100
pdm history enable
arp timeout 14400
timeout xlate 0:05:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h323 0:05:00 sip
0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
http server enable
no snmp-server location
no snmp-server contact
snmp-server community public
no snmp-server enable traps
floodguard enable
telnet timeout 5
ssh timeout 5
terminal width 80
Cryptochecksum:6380fd60019789e34aac0d059a269
: end
[OK]
PIX1(config)#
```

I put the new configurations in boldface for readability.

Do a **write mem** command to write the configuration from the running-config to the startup-config.

Network Address Translation (NAT, PAT, and Static)

Network Address Translation is a way to map a range of global addresses to an inside or perimeter (DMZ) address. This lets us hide the real IP address that we configured on the servers from the outside world. There are four flavors of NAT and they can work in several ways:

- Static NAT – maps an unregistered IP address to a registered IP address on a one-to-one basis. This is useful when you need to have an inside device accessible from the Internet.

- Dynamic NAT – maps an unregistered IP address to a pool of registered IP addresses.
- Overloading – is a form of Dynamic NAT but maps multiple unregistered IP addresses to one single registered IP address. This is also known as PAT or single address NAT.
- Overlapping – is when IP addresses on your network are registered IP address and used by another network.

Now we need to think about what to do with the other addresses that will traverse our PIX. Do we NAT them or do we have registered IP addresses and wish to use that scheme. If we have real registered IP address (global addresses) and we do not wish to hide these addresses from the world, then we could disable NAT with the NAT ID 0 command. In our case, we have private addresses behind our PIX and we wish to use NAT.

In our example we will be using single address NAT which is called PAT. PAT is where multiple internal IP addresses map or translate to one external address. Written this way it sounds as though you can only have one user connected to the Internet at a time. Well, with PAT you can indeed have multiple users connected to the Internet (theoretically 65,535). PAT works by assigning a unique TCP or UDP port number to the outside address and thus creating a uniquely numbered connection. The PIX keeps track of which outside address/port combination is assigned to which internal address. A few things to consider when using PAT are:

- The IP address used for PAT can not be used in another global pool.
- PAT does not work with H.323, PPTP, caching nameservers, or IPSEC.
- Don't use PAT with multimedia applications. The multimedia applications can conflict with the ports used by PAT.
- The IP addresses used in the global pool require reverse DNS entries to work correctly.

The PIX associates the internal addresses with the global addresses using a tag ID. For example, if the inside interface was assigned NAT ID 1 and a server from that interface wants to connect to the outside that has a global tag ID of 1, then that server is assigned an IP address from that global pool of address.

Let's form an example for our configuration. On our internal network we want users to be able to communicate with the Internet. They will be represented on the Internet by the registered IP address of 192.168.10.254, so we'll create an IP pool (even if there is only one entry in the pool). We create this pool and give it a tag number. The command to do this is:

```
PIX1(config)#global (outside) 1 192.168.10.254 netmask 255.255.255.0
```

This command states that outside interface is using an IP pool tag ID of 1. Anyone coming from a higher security level interface with a NAT tag of 1 selects an address out of the listed pool. Since there is only one IP address in the pool, all users coming from a

interface with a NAT ID of 1 will use this address. Cisco recognizes the use of only one IP address in the global pool and interprets it as a PAT statement. Next we tell the PIX which users will be using this pool. The following command accomplishing this:

```
PIX1(config)#nat (inside) 1 172.16.0.0 255.255.0.0 0 0
```

This command will assign the 172.16.0.0/16 network behind the inside interface to NAT ID 1.

The additional 0 0 's at the end of the command refer to the maximum connections and the embryonic limits. The maximum connection allows you to set the maximum number of connections and the embryonic limit allows you to specify how many "half-open" connections to allow. Setting the embryonic limit allows you to prevent certain types of attacks where a connection is started but not completed, thus leaving it "half-open". Every connection is embryonic until it sets up. These values of 0 (zero) mean "unlimited".

Another example would be:

```
PIX1(config)#global (outside) 5 192.168.10.100 – 192.168.10.200 netmask  
255.255.255.0
```

Then we could make the NAT command like this:

```
PIX1(config)#nat (inside) 5 172.16.0.0 255.255.0.0 0 0
```

This example reads: "Assign a pool of global IP addresses. Give this pool a range of addresses from 192.168.10.100 to 192.168.10.200 (100 usable addresses), and give this pool a tag ID of 5". The above NAT statement would then read: "Assign anyone on the 172.16.0.0/16 network behind the inside interface to the NAT ID of 5, and have them select an IP address from the list of IP addresses in pool ID 5 when they traverse the outside interface".

We could also assign multiple tag IDs to multiple interfaces. For example the command:

```
1 PIX1(config)#global (outside) 1 192.168.10.100-192.168.10.200 netmask  
255.255.255.0
```

```
2 PIX1(config)#global (outside) 1 192.168.10.231 netmask 255.255.255.0
```

```
3 PIX1(config)#nat (inside) 1 172.16.0.0 255.255.0.0 0 0
```

```
4 PIX1(config)#nat (dmz) 1 172.17.0.0 255.255.0.0 0 0
```

These commands are interpreted as follows:

1. Create a global pool of address from 192.168.10.100 to 192.168.10.200 (100 IP addresses) with a tag ID of 1.

2. Create a global pool of address from 192.168.10.231 (1 IP address or dynamic-NAT a.k.a PAT) with a tag ID of 1.
3. Assign the network 172.16.0.0/16 behind the inside interface the NAT tag ID 1. When users in this network traverse the outside interface, they will be translated to one of the address in the pool. If the pool is out of address to give out then they will be PATed with the address in statement 2.
4. Assign the network 172.17.0.0/16 behind the dmz interface the NAT tag ID 1. When users in this network traverse the outside interface, they will be translated to one of the address in the pool. If the pool is out of address to give out then they will be PATed with the address in statement 2.

Global's and NAT's go together. They are used in conjunction when you want to go from a higher security level interface to a lower security level interface, like from the inside to the outside. Global's are not static. This means that they will timeout and be removed from the translation table after a certain amount of time. This time is defined in the PIX configuration ([Figure 6](#)) as **timeout xlate 3:00:00** or 3 hours.

Another form of address translation is the use of static's. Static's create a permanent mapping (called a translation slot or "xlate"). They're entered into the translation table and do not expire like the global's. You would typically use a static when you are going from a lower interface to a higher one. An example of a static would be when you need to have a server on your DMZ accessible from the Internet. To accomplish this we would do two things on our PIX. First we would create the static and then we create an access-list. We'll look at the access-list command in the next section. For now the static command's syntax is:

static (*internal_if_name*,*external_if_name*) *global_ip local_ip netmask mask*

These parameters are defined as:

- *internal_if_name* is a higher security level interface name that the traffic wants to go to.
- *external_if_name* is a lower security level interface where the traffic is coming from.
- *global_ip* is the registered IP address that is accessible from the external interface.
- *local_ip* is the unregistered IP address that is the ultimate destination.
- *mask* is the subnet mask or host mask.
- There are other options and parameters for this command. For a complete definition, I suggest reading the *PIX command reference*.

The syntax of the command is a little confusing but a good way to remember this is:

Static (high.low) low high

Next following our example to have the Internet access our DMZ web server, we would create the following static command:

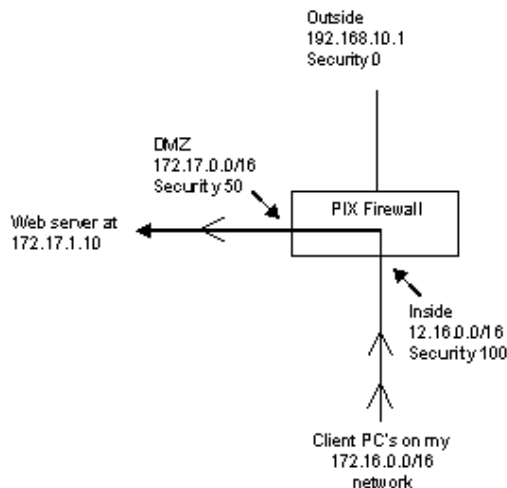
```
PIX1(config)#static (dmz,outside) 192.168.10.10 172.17.1.10 netmask
255.255.255.255 0 0
```

This command reads: “Anyone coming from the outside and trying to reach the IP address of 192.168.10.10, really send them to the server at IP address 172.17.1.10”.

This command allows the outside to connect to the server, but it doesn't tell them to which port they are allowed to connect. This is where access-lists come into play. Static's and access-lists go together. Static's translate you to the right address, whereas access-lists allow you to access services on the destination IP address specified. Another rule of traversing traffic is that interfaces with the same security level can not talk to each other.

What happens when we don't want to translate our address? But we want to traverse interfaces, keep our real IP address (static) and go from a higher to a lower interface. In other words, I am going from a high security level (inside) to a low security level (DMZ) and I want to keep my existing IP address. Let's look at [figure 7](#).

Figure 7



I have a client PC's on network 172.16.0.0 that need to go to the server at 172.17.1.10. Normally I would need a NAT and a global to go from a high to a lower security level. The PIX allows us to go to a lower security interface from a high security interface without a NAT and global command set. With no-NAT the static command takes on a little different sense of logic. With NAT disabled the PIX still allows addresses between interfaces to talk to each. In our example, we can allow the 172.16.0.0 network to talk to 172.17.1.10 without using NAT. Our static command would take on a different syntax:

```
static (high,low) high high
```

Or for our example:

```
PIX1(config)#static (inside,dmz) 172.16.0.0 172.16.0.0 netmask 255.255.0.0 0 0
```

This command reads: “When users are coming from the inside network, allow them to keep their IP address.”

On the PIX enter these commands:

```
1 PIX1(config)#global (outside) 1 192.168.10.254 netmask 255.255.255.0
2 PIX1(config)#nat (inside) 1 172.16.0.0 255.255.0.0 0 0
3 PIX1(config)#nat (dmz) 1 172.17.0.0 255.255.0.0 0 0
4 PIX1(config)#static (inside,dmz) 172.16.0.0 172.16.0.0 netmask 255.255.0.0 0 0
5 PIX1(config)#static (dmz,outside) 192.168.10.10 172.17.1.10 netmask
   255.255.255.255 0 0
6 PIX1(config)#static (dmz,outside) 192.168.10.20 172.17.1.20 netmask
   255.255.255.255 0 0
7 PIX1(config)#clear xlate
8 PIX1(config)#wr mem
```

The command explanations are as follows:

1. Create a pool of IP addresses, this pool consists of only one address (192.168.10.254), assign this pool with the tag ID of 1. Anyone using this pool will get this address, this subnet mask, and a unique port number (PAT).
2. Computers behind the inside interface and matching the network of 172.16.0.0/16 get assigned to NAT ID 1.
3. Computers behind the DMZ interface and matching the network of 172.17.0.0/16 get assigned to NAT ID 1.
4. Anyone coming from behind the inside interface and going to the DMZ interface gets to keep their current address.
5. Anyone coming from behind the outside interface and trying to connect to the external IP address of 192.168.10.10 really connect to the IP 172.17.1.10 in the DMZ. This is a one-to-one translation and requires a netmask of 255.255.255.255. Here we will also need an access-list (discussed next).
6. Anyone coming from behind the outside interface and trying to connect to the external IP address of 192.168.10.20 really connect to the IP 172.17.1.20 in the DMZ
7. Clears the translation table. This command should be run when you change or remove a nat or static command. You should be careful with this command. Running a clear xlate will clear everyone’s current translations so they will have to rebuild their connections. You can clear just a specific translation slot on the PIX. Please read the *PIX Command Reference* or visit [Cisco’s website](#).
8. Save the config with a wr mem command.

Access-Lists

Now we need to create our access-lists. Access Control Lists (ACL) are filtering rules by which the PIX will use to block or allow traffic. If there exists a valid translation for the source interface and no ACL is defined for the source interface then outbound connections are allowed by default. Said another way, IF users on the inside network in our configuration want to connect to the outside or DMZ networks AND no ACL exist on the inside network THEN they are allowed. Remember that high security levels are allowed to access lower security levels without an ACL. So the inside can connect to both the DMZ and the outside, the DMZ can connect to only the outside without ACL's. However in most cases it is advantages to restrict outbound access. Why? Suppose, for example, someone accidentally downloaded a "backdoor" virus. The virus would then install itself on your system and open ports to the outside which may allow remote control. You, in effect, just disabled your firewall.

You can restrict outbound access via the following:

- source IP address
- source port number
- destination IP address
- destination port number

If users behind the inside interface try to connect to the outside with the current configuration, they're allowed because no access-list is currently applied to the inside interface and there is a valid translation with our **global** and **nat** commands. As soon as you create an access-lists and apply it to an interface then all traffic inbound to that interface is subject to that list. PIX access-lists are only applied in the inbound direction.

For an example, suppose that we wanted to allow only IP's 172.16.1.200, and 172.16.2.201 on our inside network access to the outside for http services. We would create an access-list. The creation of an access-list is a two part process. First we create the list, and then apply or bind it to an interface. Let's first create the list:

```
PIX1(config)#access-list acl_outbound permit host 172.16.1.200 any eq www
PIX1(config)#access-list acl_outbound permit host 172.16.1.201 any eq www
PIX1(config)#access-list acl_outbound deny tcp any any eq www
```

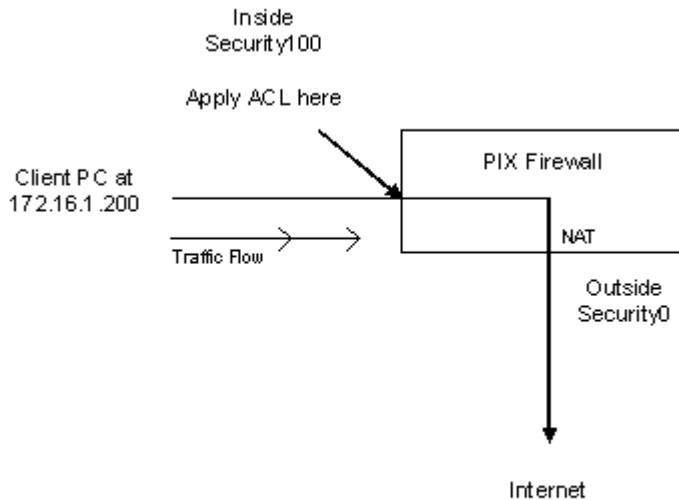
Then apply this list to the inside interface with an access-group command.

```
PIX1(config)#access-group acl_outbound in interface inside
```

As it's currently configured, only those two IP addresses can access any outside host for www services. All other IP addresses will be denied. Even the two IP's (172.16.1.200 and 172.16.1.201) will be denied if they try to use any other service besides port 80. All other IP addresses and services are denied because PIX Firewalls have an implicit deny at the end of each ACL. The PIX firewall denies all unless explicitly allowed.

Let's examine traffic as it flows through the PIX. Look at the diagram in [Figure 8](#).

Figure 8



1. Traffic at the source (172.16.1.200) is considered inbound to the PIX and checked against the ACL. Inbound traffic is from the perspective of the PIX.
2. The ACL is read in sequential order from top to bottom until it finds a rule that it matches. Once it finds the rule, it stops reading the rules and passes the traffic. If it finds no rule it drops the packet.
3. The PIX then routes the traffic to the appropriate interface.
4. The address is translated; a translation slot is created and stored in the translation table. And the packet is passed on to its destination. Another thing that happens here is that a session flow is created and added to the connection table.

The order of operations is to first check the ACL, then route, and then NAT.

We need a few access-lists on our PIX so at the config prompt type:

- ```
1 PIX1(config)#access-list acl_inside permit tcp 172.16.0.0 255.255.0.0 any eq www
2 PIX1(config)#access-list acl_inside permit tcp 172.16.0.0 255.255.0.0 any eq ftp
3 PIX1(config)#access-list acl_inside permit tcp 172.16.0.0 255.255.0.0 any eq 443
4 PIX1(config)# access-list acl_inside permit tcp 172.16.0.0 255.255.0.0 host
 172.17.1.20 eq smtp
5 PIX1(config)#access-list acl_dmz permit tcp 172.17.0.0 255.255.0.0 any eq www
6 PIX1(config)#access-list acl_dmz permit tcp host 172.17.1.10 host 172.16.1.10 eq
 1521
7 PIX1(config)#access-list acl_dmz permit tcp host 172.17.1.20 any eq smtp
```

```
8 PIX1(config)#access-list acl_outside permit tcp any host 192.168.10.20 eq smtp
9 PIX1(config)#access-list acl_outside permit tcp any host 192.168.10.10 eq www
10 PIX1(config)#access-list acl_outside permit tcp any host 192.168.10.10 eq 443
11 PIX1(config)#access-group acl_inside in interface inside
12 PIX1(config)#access-group acl_dmz in interface dmz
13 PIX1(config)#access-group acl_outside in interface outside
14 PIX1(config)#wr mem
```

The commands are interpreted as follows:

1. Allow network 172.16.0.0 (inside) to go to any host (outside or DMZ) for port www (80).
2. Allow network 172.16.0.0 (inside) to go to any host (outside or DMZ) for port ftp (21).
3. Allow network 172.16.0.0 (inside) to go to any host (outside or DMZ) for port 443 (SSL).
4. Allow network 172.16.0.0 (inside) to go to host 172.17.1.20 for SMTP (25).
5. Allow network 172.17.0.0 (DMZ) to go to any host (inside or outside) for port www (80). \*Remember just because the ACL states that we can go from the DMZ to the inside (low to high) doesn't mean that we can do it. We also have to have a valid translation in place. We do have a static in place but we have no web servers in the inside.
6. Allow DMZ host 172.17.1.10 to inside host 172.16.1.10 for 1521 (Oracle).
7. Allow network 172.17.0.0 (DMZ) to go to any host (inside or outside) for port SMTP (25).
8. Allow anyone to access host 192.168.10.20 for SMTP.
9. Allow anyone to access host 192.168.10.10 for www.
10. Allow anyone to access host 192.168.10.10 for 443 (SSL).
11. Apply ACL named acl\_inside to inbound traffic for the inside interface.
12. Apply ACL named acl\_dmz to inbound traffic for the dmz interface.
13. Apply ACL named acl\_outside to inbound traffic for the outside interface.
14. Do a write mem to save the config.

Access Control Lists have many features and extensive uses. For more information on ACL's for the PIX see the *PIX Command Reference* or visit [Cisco's website](#).

## Routing

Now we need to route. Let's again look at our network ([Figure 9](#)). For the purposes of this section I added another network behind the inside interface.



## LAN-Based Failover

You can use a pair of twin PIX devices to create a high availability solution. The Firewall's have to have to be the same model, have the same memory installed, same number of NIC's, and operation system version. Failover can be provided with no operator intervention.

One unit is the "Active" unit and the other is in "Standby" mode. The active unit performs all the normal PIX functions and the standby unit monitors the PIX's, ready to take control should the active fail.

It works like this: The active unit uses the system IP address and MAC address as the primary unit. The standby unit uses the failover IP address and the MAC address of the secondary unit. When a failover occurs the standby unit assumes the IP and MAC address of the primary, and affectively replaces each other's presence on the network. Since the active unit always uses the MAC and IP address of the primary, no ARP tables need to be updated. The units exchange message-based "hellos" that must be acknowledge (ACKed). If a message is not ACKed within a certain time frame (3 seconds) the transmission is resent. If this happens 5 times in total (15 seconds), the standby triggers a failover.

Prior to version 6.2 you had to use a special failover cable supplied with the PIX. Now in version 6.2 and above you can achieve the failover through one of the PIX interfaces configured with the **failover lan interface** *interface\_name* command. The units synch configurations. This syncing of the config is accomplished in one of three ways:

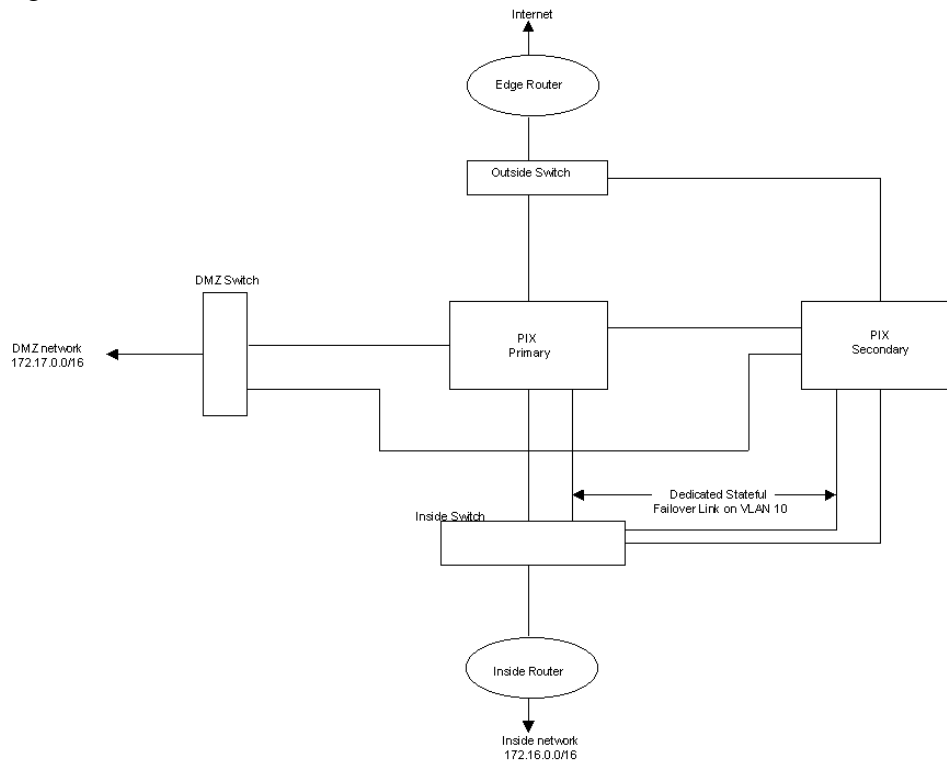
- On boot-up of the standby, the active unit will push its config to the standby.
- Commands entered on the active unit are pushed to the standby.
- When you issue a **write standby** command or a **write mem**.

When a failover occurs the connections are dropped, and they have to be reinitiated again. Starting with version 5.0 you can configure what is known as a stateful failover. A stateful failover will keep existing connections so that when a failover occurs applications will not lose connection.

To support a stateful failover, a dedicated interface on both PIX's is required. State table updates will not only be kept on the primary unit but they will also be written to the standby unit as well, in real-time. This includes xlates (static and dynamic) and connection records. This way when a failover happens connections will not be lost. It does not, however, by default transfer state information about http (80) because they are considered latency sensitive. The http connections need to be reestablished. To get around this use the **failover replicate http** command that allows for http sessions in a stateful environment. It needs to be also noted that the stateful failover link needs to be at full duplex and at least equal to the fastest speed interface on your PIX and no slower than 100full. You need to connect the PIX's together with a dedicated hub or switch, and no router can be in between them. The Stateful failover link on both sides should be on

the same VLAN. Look at [Figure 10](#) to get a picture on how we will set this up in our configuration.

Figure 10



I will use ethernet3 as the dedicated Stateful Failover Link. Now let's configure our PIX for failover. You should only have one unit powered on, if you have the other unit powered on, power it off. Remember we've already given Ethernet3 a name of SFailover with a security level of 20 and we've set its speed to 100full, but we'll do it again just for readability. From that unit type:

```

1 PIX1(config)# clock set current-time-entered-here
2 PIX1(config)#nameif ethernet3 SFailover sec20
3 PIX1(config)#interface ethernet3 100full
4 PIX1(config)#ip address SFailover 172.18.1.1 255.255.255.0
5 PIX1(config)#failover ip address SFailover 172.18.1.4
6 PIX1(config)#failover ip address outside 192.168.10.4
7 PIX1(config)#failover ip address inside 172.16.1.4
8 PIX1(config)#failover ip address dmz 172.17.1.4
9 PIX1(config)#failover lan unit primary
10 PIX1(config)#failover lan interface Sfailover
11 PIX1(config)#failover lan key shared-key-here
12 PIX1(config)#failover lan enable
13 PIX1(config)#failover active
14 PIX1(config)#wr mem

```

Now power up the second PIX, console in, get into config mode and type:

```
15 pixfirewall(config)#nameif ethernet3 SFailover sec20
16 pixfirewall(config)#interface ethernet3 100full
17 pixfirewall(config)#ip address SFailover 172.18.1.1 255.255.255.0
18 pixfirewall(config)#failover ip address SFailover 172.18.1.4
19 pixfirewall(config)#failover lan unit secondary
20 pixfirewall(config)#failover lan interface SFailover
21 pixfirewall(config)#failover lan key shared-key-here
22 pixfirewall(config)#failover lan enable
23 pixfirewall(config)#failover
24 pixfirewall(config)#wr mem
```

Power off the secondary unit then power it back on.

These commands are interpreted as:

1. Sets the clock on the primary so when the secondary comes on line the two PIX's time will match.
2. Give a name to ethernet3 and set its security level
3. Set the speed and duplex of ethernet3, and turns it on.
4. Give Ethernet3 an IP address.
5. Failover IP address for this interface.
6. Failover IP address for this interface.
7. Failover IP address for this interface.
8. Failover IP address for this interface.
9. Makes this unit primary.
10. This link will be used for LAN failover.
11. Sets up a pre-shared key.
12. Enables failover.
13. Makes this unit the active unit.
14. Saves the config on the primary.

On secondary:

15. Give the same name to ethernet3 and set its security level the same.
16. Set the speed and duplex of ethernet3 to the same as the primary and turn it on.
17. Give Ethernet3 on the secondary the same IP address as in step 4.
18. Failover IP address for this interface
19. Makes this unit secondary.
20. This link will be used for LAN failover.
21. Sets up a pre-shared key.
22. Enables failover.
23. Makes this unit the secondary unit because the active keyword is not used.
24. Saves the config.

Your systems are now setup for stateful failover.

## Management

The most common way to administer the PIX is via telnet. But by default the PIX blocks telnet access. You must go into config mode and add those PC IP addresses or networks that are allowed to telnet into the PIX. You will also need to tell the PIX which interface can accept telnet traffic.

```
PIX1(confi)#telnet 172.16.0.0 255.255.0.0 inside
```

This command reads: “Allow telnet access for network 172.16.0.0/16 to the inside interfaces IP address.”

Another popular way to administer the PIX is thru the PIX Device Manager. PDM is a GUI/Wizard interface into the PIX. PDM runs on software version 6.0 or greater. It allows you to do anything to the PIX that you can do with the command line interface (CLI). PDM also:

- Runs in a browser via Java.
- Operates on any PIX.
- Does not require a plug-in software installation.
- Comes preloaded on new firewalls with code 6.0 and above.
- Works with SSL for secure communications.

With the System Properties section of the PDM interface, you can configure failover, logging, AAA, and DHCP servers as well as other features. The PDM also comes with a scaled down version of an Intrusion Detection System (IDS). With PDM version 2.0 you can setup a user database and then restrict user access capabilities.

## Conclusions

With this configuration you have a functional Cisco PIX Firewall that offers high availability and manageability in a secure manner. There are a lot of things that I did not discuss in this document; like VPN, the fixup protocol, logging, monitoring, and AAA. There were also many options for commands that I did not mention. I just wanted to provide a document that the reader could use to setup a PIX with a basic configuration. I suggest visiting Cisco’s Website and reading the literature about command references. There are not many books on the PIX but here is one: [Cisco Secure PIX Firewalls](#), by David W. Chapman Jr., Andy Fox, Cisco Press. I also did not go over creating security policies or IDS systems. This document is just scratching the surface of what needs to be

accomplished when securing an enterprise, but understanding, troubleshooting and configure a PIX Firewall will go a long way to completing that journey.

## Citations

“Cisco PIX Firewall and VPN Configuration Guide, Version 6.2.”, Cisco Website (2002 Cisco System Inc.)

URL: [http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix\\_62/config/](http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_62/config/)

“Cisco PIX Firewall Command Reference, Version 6.2”, Cisco Website (2002 Cisco System Inc.)

URL: [http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix\\_62/cmdref/](http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_62/cmdref/)

“Cisco’s PIX Firewall Series and Stateful Firewall Security.”, Cisco Website (2000 Cisco Systems Inc.)

URL: [http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/tech/nat\\_wp.htm](http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/tech/nat_wp.htm)

“Cisco Secure Device Manager”, Cisco Website (2001 Cisco Systems Inc.)

URL: [http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/prodlit/pixdm\\_ds.htm](http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/prodlit/pixdm_ds.htm)

Howard, Jeffery, “Packet Filters, Stateful Packet Filters, and Proxies.” Infrequently Asked Questions.

URL: <http://www.burningvoid.com/iaq/firewall-type.html>

“How Failover Works on the Cisco Secure PIX Firewall.”, Cisco Website (2002 Cisco Systems Inc.)

URL: <http://www.cisco.com/warp/public/110/failover.html>

How Nat Works, Cisco Website (2001 Cisco Systems Inc.)

URL: <http://www.cisco.com/warp/public/556/nat-cisco.shtml>

“NAT Order of Operations.”, Cisco Website (2002 Cisco Systems Inc.)

URL: <http://www.cisco.com/warp/public/556/5.html>

Spitzner, Lance, “Understanding the FW-1 State Table.” 29 November, 2000.

URL: <http://www.enteract.com/~lspitz/fwtable.html>

“Using NAT, Global, Static, Conduit, and Access-List Commands and Port Redirection on the PIX.”, Cisco Website (2002 Cisco Systems Inc.)

URL: <http://www.cisco.com/warp/public/707/28.html>

Welcher, Peter J. and Moerschel, Grant, “Cisco Firewall Basics” Ciscoworld. April 2002: 17 – 19.

URL: <http://www.ciscoworldmagazine.com/monthly/2001/04/firewall.shtml>

## Appendix – A

Listed is the complete primary PIX configuration. ([Figure 11](#))

Figure 11

```
PIX1(config)# wr t
Building configuration...
: Saved
:
PIX Version 6.2
nameif ethernet0 outside security0
nameif ethernet2 DMZ security50
nameif ethernet3 SFailover security20
nameif ethernet1 inside security100
enable password xxxxxxxxxxxxxxx encrypted
passwd xxxxxxxxxxxxxxx encrypted
hostname PIX1
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
names
no pager
logging on
no logging timestamp
no logging standby
no logging console
no logging monitor
no logging buffered
no logging trap
no logging history
logging facility 20
logging queue 512
interface ethernet0 100full
interface ethernet1 100full
interface ethernet2 100full
interface ethernet3 100full
mtu outside 1500
mtu inside 1500
```

mtu dmz 1500  
mtu sfailover 1500  
**ip address dmz 172.17.1.1 255.255.0.0**  
**ip address inside 172.16.1.1 255.255.0.0**  
**ip address outside 192.168.10.1 255.255.255.0**  
**ip address sfailover 172.18.1.1 255.255.255.0**  
ip audit info action alarm  
ip audit attack action alarm  
**failover ip address SFailover 172.18.1.4**  
**failover ip address outside 192.168.10.4**  
**failover ip address inside 172.16.1.4**  
**failover ip address dmz 172.17.1.4**  
**failover lan unit primary**  
**failover lan interface Sfailover**  
**failover lan key *shared-key-here***  
**failover lan enable**  
**failover active**  
pdm logging informational 100  
pdm history enable  
arp timeout 14400  
**nat (inside) 1 172.16.0.0 255.255.0.0 0 0**  
**nat (dmz) 1 172.17.0.0 255.255.0.0 0 0**  
**global (outside) 1 192.168.10.254 netmask 255.255.255.0**  
**static (inside,dmz) 172.16.0.0 172.16.0.0 netmask 255.255.0.0 0 0**  
**static (dmz,outside) 192.168.10.10 172.17.1.10 netmask 255.255.255.255 0 0**  
**static (dmz,outside) 192.168.10.20 172.17.1.20 netmask 255.255.255.255 0 0**  
**access-list acl\_inside permit tcp 172.16.0.0 255.255.0.0 any eq www**  
**access-list acl\_inside permit tcp 172.16.0.0 255.255.0.0 any eq ftp**  
**access-list acl\_inside permit tcp 172.16.0.0 255.255.0.0 any eq 443**  
**access-list acl\_inside permit tcp 172.16.0.0 255.255.0.0 host 172.17.1.20 eq smtp**  
**access-list acl\_dmz permit tcp 172.17.0.0 255.255.0.0 any eq www**  
**access-list acl\_dmz permit tcp host 172.17.1.10 host 172.16.1.10 eq 1521**  
**access-list acl\_dmz permit tcp host 172.17.1.20 any eq smtp**  
**access-list acl\_outside permit tcp any host 192.168.10.20 eq smtp**  
**access-list acl\_outside permit tcp any host 192.168.10.10 eq www**  
**access-list acl\_outside permit tcp any host 192.168.10.10 eq 443**  
**access-group acl\_inside in interface inside**  
**access-group acl\_dmz in interface dmz**  
**access-group acl\_outside in interface outside**  
**route inside 172.16.0.0 255.255.255.0 172.16.1.2 1**  
**route inside 172.20.0.0 255.255.255.0 172.16.1.2 1**  
**route outside 0 0 192.168.10.2 1**  
timeout xlate 0:05:00  
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h323 0:05:00 sip  
0:30:00 sip\_media 0:02:00  
timeout uauth 0:05:00 absolute

```
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
http server enable
no snmp-server location
no snmp-server contact
snmp-server community public
no snmp-server enable traps
floodguard enable
telnet 172.16.0.0 255.255.0.0 inside
telnet timeout 5
ssh timeout 5
terminal width 80
Cryptochecksum:6380fd60019789e34aac0d059a269
: end
[OK]
PIX1(config)#
```

© SANS Institute 2002, Author retains full rights.



# Upcoming SANS Training

[Click Here for a full list of all Upcoming SANS Events by Location](#)

|                                                                      |                        |                             |            |
|----------------------------------------------------------------------|------------------------|-----------------------------|------------|
| Hong Kong Advanced Forensics Seminar                                 | Hong Kong, Hong Kong   | Nov 09, 2009 - Nov 14, 2009 | Live Event |
| SANS Sydney 2009                                                     | Sydney, Australia      | Nov 09, 2009 - Nov 14, 2009 | Live Event |
| SANS Vancouver 2009                                                  | Vancouver,             | Nov 14, 2009 - Nov 19, 2009 | Live Event |
| SecurityByte 2009                                                    | New Delhi, India       | Nov 17, 2009 - Nov 20, 2009 | Live Event |
| SANS Geneva CISSP at HEG 2009 Autumn                                 | Geneva, Switzerland    | Nov 23, 2009 - Nov 28, 2009 | Live Event |
| SANS London 2009                                                     | London, United Kingdom | Nov 28, 2009 - Dec 06, 2009 | Live Event |
| SANS WhatWorks in Incident Detection Summit 2009                     | Washington, DC         | Dec 09, 2009 - Dec 10, 2009 | Live Event |
| SANS CDI East 2009                                                   | Washington, DC         | Dec 11, 2009 - Dec 18, 2009 | Live Event |
| SANS WhatWorks in Data Leakage Prevention and Encryption Summit 2010 | New Orleans, LA        | Jan 07, 2010 - Jan 12, 2010 | Live Event |
| SANS Security East 2010                                              | New Orleans, LA        | Jan 10, 2010 - Jan 18, 2010 | Live Event |
| SANS AppSec 2010 and WhatWorks in AppSec Summit                      | San Francisco, CA      | Jan 29, 2010 - Feb 05, 2010 | Live Event |
| SANS San Francisco 2009                                              | OnlineCA               | Nov 09, 2009 - Nov 14, 2009 | Live Event |
| SANS OnDemand                                                        | Books & MP3s Only      | Anytime                     | Self Paced |