Advanced Protocol Handling and PIX Firewall Features

Terms you'll need to understand:

- ✓ Fixups
- ✓ Standard and passive FTP
- ✓ SCCP
- ✓ Skinny
- ✓ SIP
- ✓ H.323
- ✓ URL filtering
- ✓ PPPoE
- ✓ Default route

Techniques you'll need to master:

- ✓ Using the Fixup protocol
- ✓ Configuring URL filtering
- ✓ Monitoring URL filtering
- ✓ Using DCHP servers and clients
- ✓ Configuring PPPoE
- ✓ Setting default routes

Firewalls today have to be very sophisticated about how to control traffic. Basic traffic can flow through a firewall effortlessly, but complex protocols need to have extra help. Cisco provides the PIX firewall with fixup support to assist these complex protocols. This chapter explains what they do, when they are required, and how to configure them. In addition, it covers the advanced topics of content filtering, DHCP settings, PPPoE configuration, and Routing Information Protocol (RIP) support.

Problems with Advanced Protocols and ASA

Several advanced protocols, including FTP, cause problems when trying to traverse across the PIX firewall. The problems arise when traffic on the outside client or server wants to send traffic to the inside, higher-security interfaces; this traffic is often unsolicited from the perspective of standard ASA. Normally, traffic flow is in response to a client's request and returns on the same source port on which the client request was sent. The ASA sees this normal request and opens a connection slot for the return traffic. Some advanced protocols respond or send data to the client on port numbers other than the ports in the source header, and this causes a problem for the normal ASA engine.

For example, if Jack is trying to download information from an FTP site using standard mode, he notifies the FTP server that his port—for example, 3002—is available to receive the data. The requested port 3002 is not in the normal source port header location but in the data portion of the packet. Because the ASA normally monitors the source port header and not the data portion, the connection slot is not made. As the FTP server starts to send data to Jack's port (3002), the PIX drops the packets because ASA never created a connection slot for the returning traffic.

The Function of Fixups

The PIX firewall implements fixup protocol features to help overcome the difficulties with advanced protocols. The fixup protocols perform what is known as *application inspection* on a limited number of advanced protocols. The inspection monitors the traffic across the PIX and dynamically opens and closes connection slots between the inside and outside interfaces. Fixups try to make the connections as secure as possible by dynamically opening only the necessary ports.

If fixup protocols did not exist, you would have to open large numbers of ports with ACLs or the established commands to allow traffic to pass, effectively compromising the granularity and overall value of your security solution. Table 8.1 displays some of the available fixup protocols with their respective ports and functions.

Table 8.1 Available Fixup Protocols

Protocol	Default Port	Function
FTP	21	The FTP fixup works to help correct standard and passive FTP problems.
H323 h225	1720	The H323 monitors and helps correct the multimedia applications that use H323 back through the PIX firewall.
H323 RAS	1718 and 1719	This works with the H323 protocol suite.
HTTP	80	Helps monitor HTTP and is required for WebSense or N2H2 URL filtering services.
ILS	389	The ILS fixup works to help correct LDAP transactions across the PIX firewall.
RSH	514	Remote Shell.
RTSP	554	Real-Time Streaming Protocol.
SMTP	25	Simple Mail Transport Protocol.
SQL*Net	1521	Oracle communications.
SIP	5060	Session Initiation Protocol.
SCCP	2000	Skinny Client Control Protocol.

The show fixup Command

You can use the `show fixup` command to display the active fixup protocols on the PIX firewall. Listing 8.1 displays the output of the `show fixup` command.

Listing 8.1 show fixup Command Example

```
pixfirewall(config)# show fixup
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol ils 389
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
pixfirewall(config)#
```

The fixup protocol Command

The standard `fixup` command is similar for all the protocols listed in Table 8.1. Most protocols can have additional ports assigned to them that will enable application inspection monitoring of nonstandard ports for that protocol. This is the standard `fixup protocol` command's syntax:

```
pixfirewall(config)# [no] fixup protocol <prot> [<option>] <port>[-<port>]
```

Table 8.2 displays the `fixup protocol` options.

Table 8.2 fixup protocol Command Options

Option	Function
prot	Protocol setting, such as HTTP, SIP, RTSP, and so on.
port-port	A single port or a range or ports can be used to enable application inspections on traffic defined for the protocol option.

The following example adds a single port and a range of nonstandard ports for RTSP:

```
pixfirewall(config)# fixup protocol rtsp 1501
pixfirewall(config)# fixup protocol rtsp 1700-1710
pixfirewall(config)# show fixup protocol rtsp
fixup protocol rtsp 554
fixup protocol rtsp 1501
fixup protocol rtsp 1700-1710
pixfirewall(config)#
```

The clear fixup Command

The `clear fixup` command resets the fixup protocol to the default values, like so:

```
pixfirewall(config)# clear fixup
```

The File Transfer Protocol

File Transfer Protocol (FTP) enables two computers to upload and download data across a network. Although it has been around for a long time, as has Telnet and email, it is considered an advanced protocol because it operates a little differently in the way it uses ports.

FTP uses two main ports—20 and 21. Port 21 is used for a control connection that is used to transmit commands to and from the FTP server. For example, as a user enters FTP commands, the commands are transmitted on

port 21. When data must be downloaded, port 20 provides this basic function. FTP comes in two main modes: standard and passive.

If you want to prevent FTP traffic, you need to block only one port. By blocking port 21, you prevent FTP commands from being sent to the normal default FTP servers. Port 20 doesn't need to be blocked because, without the commands, data can't be transferred.

Standard Mode

FTP operates in a couple of modes. In *standard* mode, the FTP client and the server send commands across a command connection on port 21. In this command connection, the client requests to use a port for uploading or downloading. This request is embedded inside the data portion of a packet sent to the server. Because the ASA monitors source port and destination port headers, this request is missed by ASA. Additionally, as the server initiates the data connection back to the client, the firewall drops the packets because no connection slot is created for this traffic.

Figure 8.1 show a basic example of a client requesting traffic from an FTP server. In step 1 the client requests to use 3002 as its data port; in step 2 the server starts to make a connection to that port. No connection slot for port 3002 exists, so the packets are dropped.

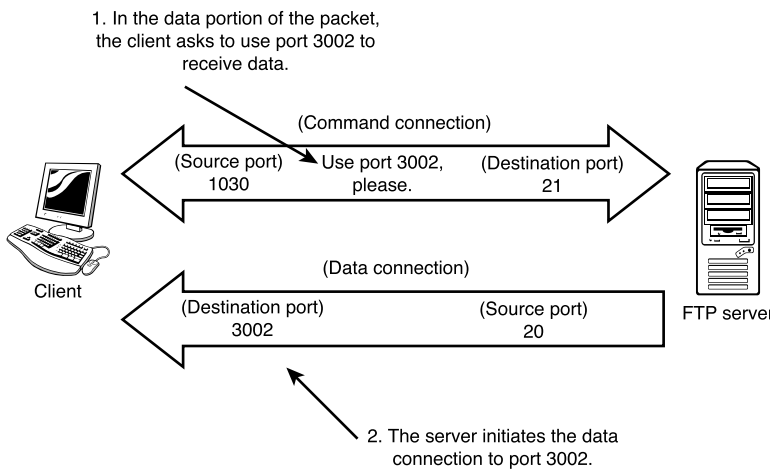

Figure 8.1 Standard mode FTP.

Passive Mode

The second FTP mode is *passive* mode, which operates a little differently from standard mode. The command connection still exists using port 21 on the server. However, the data connection on the server doesn't have to use port 20. When data needs to be transmitted, the client requests asks the server which port it should use. Again, this request is embedded in the data portion of a packet within the command connection traffic. The server sends the port number it wants the client to use, and then the client initiates the data connection to the server. This is the exact opposite from standard mode, in which the server initiates the data connection.

Figure 8.2 is a basic example of passive mode FTP. In step 1 the client and server negotiate which server port will be used to transfer data; then in step 2 the client initiates the data connection to the server. Because the client makes this connection, the ASA creates a connection slot and has no problems allowing traffic to pass back and forth.

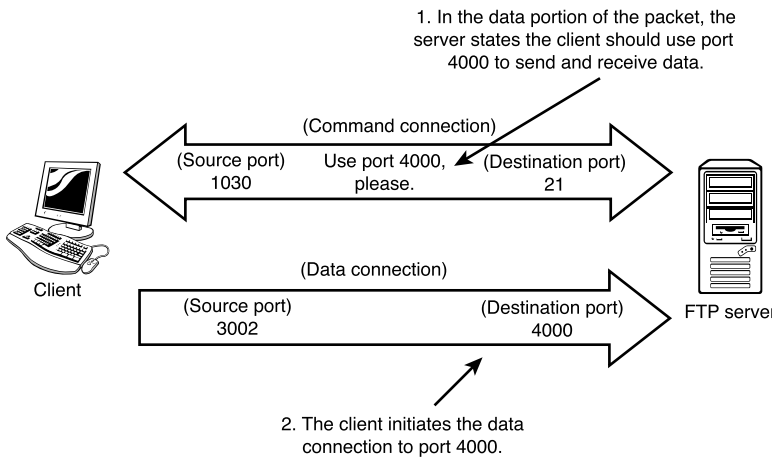

Figure 8.2 Passive mode FTP.

It is important that you understand the difference between standard and passive mode. A good way to remember the difference is as follows:

In *standard* mode the server calls the shots and initiates the data connection. This means clients on the inside have trouble connecting in standard mode.

In *passive* mode, the server is passive and the client initiates the data connections. This means that passive mode works very well for clients on the inside of the PIX firewall.

The fixup protocol ftp Command

With the possible problems of FTP data connections and the ASA dropping uninitiated traffic, the PIX has a fixup protocol option for FTP that compensates for the FTP traffic. This fixup monitors the embedded data portions of the command connection traffic. When embedded port requests are detected, the PIX dynamically creates connection slots for the necessary ports, allowing the uninitiated traffic to flow. The following is the syntax for the `ftp` command:

```
pixfirewall(config)# fixup protocol ftp <port> [<strict>]
```

Table 8.3 fixup protocol ftp Command Options

Option	Function
port	This is the port number to monitor for FTP traffic. Typically, this is port 21.
strict	The strict option prevents any embedded FTP commands in HTTP connections. By default, this is allowed.

The following is the command to enable the FTP fixup protocol:

```
pixfirewall(config)# fixup protocol ftp 21
```

The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) fixup protocol allows application inspection of traffic using the default port 80. When HTTP fixups are enabled, three main functions become available:

- Filtering of URL features using WebSense or N2H2 servers
- Logging of HTTP `GET` requests
- Filtering of Java and ActiveX content.

You can create additional ports for the HTTP fixup protocol command. For example, the following command enables HTTP fixups on port 8080:

```
pixfirewall(config)# fixup protocol http 8080
```

With the previous command, application inspection for HTTP takes place on the both the default port 80 and on additional port 8080.

To turn off HTTP fixups, you use the following command:

```
pixfirewall(config)# no fixup protocol http 8080
pixfirewall(config)# no fixup protocol http 80
```

If you turn off all HTTP fixups, you will not able to perform URL filtering.

Remote Shell

Remote Shell (RSH) was originally created for Unix systems as an easy-to-use remote console that doesn't need a login as its brother Telnet does. RSH is very insecure and should be replaced at all costs with more secure connections, such as SSH.

RSH is similar to standard mode in the FTP protocol. Two connections are required for complete communication—one connection for commands and a second for standard error outputs. The client embeds the port number to which the server should send standard errors. The server then initiates the second connection that will not be in the connection table. If the `fixup protocol rsh` command is not enabled, the ASA rejects the server's request.

The `fixup protocol rsh` command inspects the RSH traffic for the embedded port requests needed on port 514. When a request for a port is sent, the ASA dynamically creates a connection slot to allow the server to send traffic back to the client. The following is the command syntax for RSH:

```
pixfirewall(config)# [no] fixup protocol rsh <port-[port]>
```

The following example enables RSH inspection on a range of additional ports:

```
pixfirewall(config)# fixup protocol rsh 2000-2003
```

SQL*Net Protocol

The SQL*Net protocol is used by Oracle clients and servers to query SQL databases. This advanced protocol mainly uses a single port for communication and should therefore not be an issue for the ASA engine. However, that port can be redirected to a different port or a different server during the connection lifetime. To allow traffic to pass securely, the `fixup protocol sqlnet` command is used to help monitor SQL*Net protocol connections. The following is the syntax for this command:

```
pixfirewall(config)# [no] fixup protocol sqlnet <port-[port]>
```

The following example enables SQL*Net inspection of port 1521:

```
pixfirewall(config)# fixup protocol sqlnet 1521
```

The Real Time Streaming Protocol

The Real Time Streaming Protocol (RTSP) is a real-time audio and video protocol used by several multimedia applications, such as RealPlayer, Cisco IP/TV, Quicktime 4, Netshow, and VDO live, to name a few. Similar to FTP, this protocol can operate in different modes depending on the application used. Each mode uses three different connections to function properly.

Real-Time Protocol (RTP) mode uses the following three connections:

- TCP control channel
- UDP RTP data channel
- UDP RTCP reports

RealNetworks' Real Data Transport (RDT) mode uses the following three connections:

- TCP control channel
- UDP data channel
- UDP resend

By default, no fixup protocols are set for RTSP applications; they must be manually set if such applications are needed. The default port used by RTSP is port 554. The following command enables fixups for RTSP:

```
pixfirewall(config)# fixup protocol rtsp 554
```

RTSP is not supported using PAT. Most RTSP applications are also incompatible with NAT.

Voice Over IP

Voice over IP (VoIP) is not one protocol but a term used for several types of protocols that provide telephone call–like connections across IP networks. Protocols such as SCCP, SIP, and H.323 are covered here.

The Skinny Client Control Protocol

Skinny Client Control Protocol (SCCP) is typically just called *Skinny*. Cisco uses this simplified protocol for its VoIP phones and CallManager servers.

The basis of Skinny is its interoperability with another protocol called H.323, which is discussed later.

When an IP phone first boots, it requests an IP address from a DHCP server. Then, the phone downloads its configuration from a TFTP server and is ready for use.

When a call is made, the client's phone sends a signal connection to a CallManager server, which then contacts the destination phone and acquires the UDP port the phone needs for audio communication. Next, the server passes this information back to the calling phone so that the source and destination phones can connect. Figure 8.3 shows that basic high-level flow for a connection.

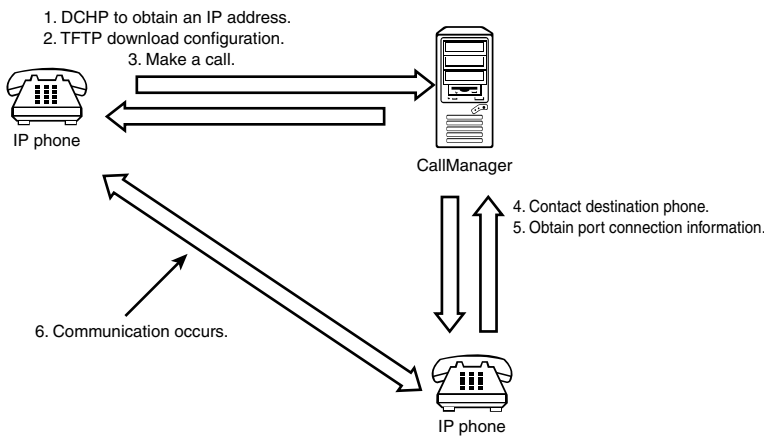

Figure 8.3 Basic Skinny VoIP flow.

Some of the application inspection problems with Skinny include the use of inside addresses and the dynamic destination port numbers. If the clients are behind NAT, the embedded IP information must be changed to reflect the external Internet addresses. The fixup protocol for Skinny monitors and changes the internal address used by NAT to an external address. It also dynamically creates connection slots to allow traffic to pass as needed.

Skinny is supported on NAT but not PAT.

The following command enables the SCCP protocol to function across the PIX firewall:

```
pixfirewall(config)# fixup protocol skinny 2000
```

The Session Initiation Protocol

Session Initiation Protocol (SIP) is another VoIP protocol that allows connections between audio devices using IP. This protocol is similar to Skinny; at a high level, the process of making a call is the same. The caller contacts what is known as a VoIP *gateway*. This gateway locates the destination phone for the caller and helps the two get connected.

The default port for VoIP gateways is UDP port 5060. The following command enables SIP fixups:

```
pixfirewall(config)# fixup protocol sip 5060
```

H.323

H.323 is a complicated hybrid protocol that can be used for VoIP, video, and data. Like other multimedia protocols, but unlike VoIP, H.323 requires several ports to connect two devices. The protocol is actually a suite of other protocols put together to make the connections desired.

The following lists standards used between two H.323 devices:

- H.225 Registration, Admission, and Status (RAS)
- H.235 Call Signaling
- H.245 Control Signaling
- TPKT Header
- Q.931 Messages
- ASN.1 Encoding Packets

Several vendors use H.323 for their products; however, each vender implements this protocol in a slightly different way. So, not all H.323 applications are supported on the PIX firewall. The following is list of supported H.323 applications:

- Cisco Multimedia Conference Manager
- Microsoft NetMeeting
- CUseeMe Meeting Point and Pro
- Intel Video Phone
- VocalTec Internet Phone and Gatekeeper

Each of these applications needs its own special adjustments for available ports. See Cisco's Web site for detailed configurations needed for each application. Here are variations of the basic command to enable H.323:

```
pixfirewall(config)# fixup protocol h323 1720
pixfirewall(config)# fixup protocol h323 1718-1719
```

Web Traffic Filtering

Although the firewall's main purpose is to protect inside users from outside threats, the PIX firewall can also help control which Web sites internal users can access. The PIX firewall can be linked to a URL filtering server such as WebSense or N2H2, which provide Internet monitoring and URL Web site blocking if necessary.

Figure 8.4 displays the basic Web filter process, which includes these steps:

1. The client opens a connection to a Web server and sends an HTTP `GET` message to access a Web page.
2. The PIX intercepts the call and forwards the request to the URL filtering server and the Web site at the same time.
3. The filtering server searches its database of Web sites to see whether the user has permission to access the Web site. In the meantime, the Web site is attempting to respond to the user's request.
4. If the URL server's response is yes, the PIX allows the Web site response to be forwarded to the requesting client. Otherwise, the Web site's response is dropped.

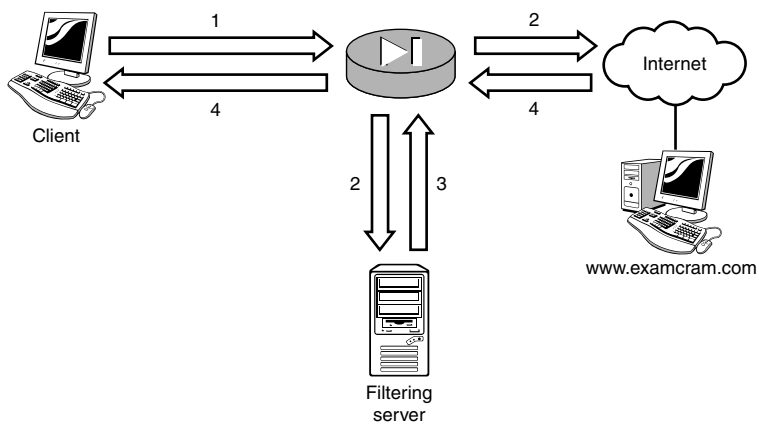

Figure 8.4 URL filtering process.

Configuring WebSense and N2H2

The PIX firewall can be configured to use WebSense or N2H2 URL filtering servers for HTTP traffic. The basic steps are the same for each vendor's configuration on the PIX firewall. They are as follows:

1. Identify the URL filtering server.
2. Specify which traffic needs to be forwarded to the filtering server.
3. Optionally, configure the URL cache.

HTTP fixup protocols must be configured to allow URL filtering.

The url-server Command

The first step is to identify the URL server you want to use. You use two different commands, based on which vendors you are actually using. However, their basic structures are the same: Identify the interface, identify the host, and set the timeout durations. The command syntax is as follows:

```
pixfirewall(config)# [no] url-server [<(if_name)>] [vendor websense]
                host <local_ip> [timeout <seconds>]
                [protocol TCP¦UDP [version 1¦4]]
pixfirewall(config)#[no] url-server [<(if_name)>] vendor n2h2
                host <local_ip> [port <number>]
                [timeout <seconds>] [protocol TCP¦UDP]
```

The filter url Command

After the servers have been configured, you need to specify which traffic will be forwarded to them. The `filter URL` command is used to identify which local users' Web traffic will be forwarded to the URL servers. The following displays the syntax needed:

```
pixfirewall(config)# [no] filter url <port>[-<port>]¦except
                <lcl_ip> <mask> <frgn_ip> <mask> [allow]
```

The `filter url` command allows you to be granular enough to select specific inside (`lcl_ip`) to outside (`frgn_ip`) ranges to filter. The `except` option enables you to exclude certain IP addresses from the filter. The `allow` option defines what the PIX firewall will do when WebSense or N2H2 servers are offline. If `allow` is stated, Web traffic is allowed to pass through the firewall. Conversely, if `allow` is not stated, all Web traffic is blocked.

The url-cache Command

Web filtering does come at a cost to performance. The delays introduced by querying an external URL filtering server can be an issue. By using the `url-cache` command, the PIX can cache a request locally on the firewall and reuse this cache the next time a user goes to the same destination. This decreases the impact of delays and increases the users' throughput. However, you do lose some tracking information about users' Web activity that would have been recorded on the URL server. The command syntax for the `url-cache` command is shown here:

```
pixfirewall(config)# [no] url-cache <dst¦src_dst> size <Kbytes>
```

A URL Filtering Example

The following sequence of commands configures a WebSense filter as the filtering server. It then specifies that all traffic is to be forwarded to the WebSense filter, except traffic initiated by 192.168.1.11:

```
pixfirewall(config)# url-server (inside) vendor websense
               host 192.168.1.101 timeout 5 protocol TCP version 4
pixfirewall(config)# filter url http 0 0 0 0
pixfirewall(config)# filter url except 192.168.1.11 255.255.255.255
               0 0 allow
pixfirewall(config)# url-cache dst 128
```

Monitoring URL Filtering

Several commands allow you to view your configuration and monitor your URL filtering traffic. Table 8.4 lists several of these.

Table 8.4 URL Filtering Commands

Command	Function
show url-cache stat	Displays URL cache details
show url-server	Displays the list of URL servers configured
show filter	Displays the URL filters configured
show perfmon	Displays performance monitor statistics, including URL access information

To view URL filtering statistics, you use the **show perfmon** and show **url-cache stat** commands.

Filtering Java Applets and ActiveX Scripts

Web pages can use powerful features such as Java applets and ActiveX scripts. The scripts enable Web developers to provide dynamic content on Web pages. In the wrong hands, however, these scripts can be created to cause harm or collect considerable information about your computer's Internet browsing history. Most browsers enable you to control your security setting for scripts. But in a very secure environment, you might need to ensure that none of these scripts can be executed after traveling across the firewall to the inside users. Cisco has two commands that enable you to comment out the scripts in the HTTP Web pages before they reach clients' computers.

The filter java Command

The `filter java` command is a new command that allows you to specify which internal and external traffic should be filtered for Java code. The filtering adds comment tags around the Java code in the Web page. These comment tags prevent the scripts from being executed. The following is the command syntax:

```
pixfirewall(config)# [no] filter Java <port>[-<port>] <lcl_ip> <mask>
                <frgn_ip> <mask>
```

`lcl_ip` specifies the internal IP address (local), and `frgn_ip` specifies the external IP address (foreign) you want to filter. The following example filters Java code for all users to all Web site IP addresses:

```
pixfirewall(config)# filter Java http 0 0 0 0
```

The filter activex Command

ActiveX scripts can also be filtered in the same way that Java scripts can. The command is basically the same:

```
pixfirewall(config)# [no] filter ActiveX <port>[-<port>]
                <lcl_ip> <mask> <frgn_ip> <mask>
```

The following command filters ActiveX content:

```
pixfirewall(config)# filter ActiveX http 0 0 0 0
```

The Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol (DHCP) allows computers to obtain IP addresses and network configurations automatically from a DHCP server. The PIX firewall can be both a DHCP client on the outside interface and at the same time provide DHCP server functionality on the inside interface.

The PIX firewall can be a DHCP client and a DHCP server at the same time.

DHCP Clients

The PIX firewall can be a DHCP client on the outside interface, enabling you to receive IP address and configuration information dynamically from another source such as an Internet service provider (ISP). The following is the command syntax:

```
pixfirewall(config)# ip address <if_name> dhcp [setroute]
```

The `dhcp` option is used with the `ip address` command to enable the interface to dynamically receive an IP address from a DHCP server source. The following example defines the outside to be a DHCP client rather than to use a fixed address:

```
pixfirewall(config)# ip address outside dhcp setroute retry 4
```

The `setroute` option enables you to receive the default route from the DHCP server, whereas the `retry` option enables the PIX to retry contacting the DHCP server a number of times before giving up. To renew your lease, you type the IP address command again.

After you have received an address, you can use the `show IP address outside dhcp` command to display the configuration information received.

DHCP Servers

The PIX can also perform the functions of a small DHCP server. The number of clients it can support is limited, and performing this function is really intended only for small SOHO environments. To configure the PIX to be a DHCP server, the commands in Table 8.5 are available.

Table 8.5 DHCP Server Commands

Command	Function
dhcpd address <ip1>[-<ip2>] inside	This sets the pool of addresses the server will hand out to clients.
dhcpd ping_timeout <timeout>	This command is the response delay the PIX uses as it tests for any other clients that might be using the address it currently wants to give a client.
dhcpd auto_config [<clnt_ifc_name>]	This command forwards all the options learned from the outside interface to the inside users.
dhcpd domain <domain_name>	This specifies the domain option.
dhcpd dns <dnsip1> [<dnsip2>]	This allows you to enter two DNS server IP addresses.
dhcpd wins <winsip1> [<winsip2>]	This allows you to enter two WINS server IP addresses.
dhcpd lease <lease_length>	This is the duration of the lease that clients will keep addresses before returning to the server for a new one.
dchpd option	This allows you to specify any additional options that might be needed.

Listings 8.2 and 8.3 show examples that configure the PIX to hand out IP addresses in the range of 192.168.1.2–192.168.1.33 with options manually configured or with options automatically configured. Automatic configuration allows the options learned from the outside DHCP server to be used as the default options for the inside clients.

Listing 8.2 demonstrates how to configure the PIX as a DHCP server with manually configured options to give to DHCP clients.

Listing 8.2 Configuring a DHCP Server with Manual Options

```
pixfirewall(config)# dhcpd address 192.168.1.2-192.168.1.33 inside
pixfirewall(config)# dhcpd lease 3000
pixfirewall(config)# dhcpd dns 192.168.1.100 192.168.1.101
pixfirewall(config)# dhcpd wins 192.168.1.99
pixfirewall(config)# dhcpd domain examcram.com
pixfirewall(config)# dhcpd enable
```

The **dhcpd dns** command allows you to set only two DNS server IP addresses.

Listing 8.3 demonstrates how to configure the PIX as a DHCP server with automatically configured options that are originally received from the ISP and are passed on to the PIX DHCP clients.

Listing 8.3 Configuring a DHCP Server with Automatic Options

```
pixfirewall(config)# dhcpd address 192.168.1.2-192.168.1.33 inside
pixfirewall(config)# dhcpd lease 3000
pixfirewall(config)# dhcpd auto_config
pixfirewall(config)# dhcpd enable
```

To display DHCP settings and bindings, the commands in Table 8.6 can be used.

Table 8.6 show dhcp Commands

Command	Function
show dhcpd	Displays current DHCP server settings
show dhcpd binding	Displays the MAC address-to-IP address bindings the PIX has assigned
show dhcpd statistics	Shows the active IP address leases, expired bindings, and several other extensive DHCP server details

The PIX firewall automatically issues the inside interfaces IP address as the default gateway option to the DHCP clients.

The Point-to-Point Protocol over Ethernet

The Point-to-Point Protocol over Ethernet (PPPoE) is an ethernet encapsulation of the Point-to-Point Protocol used most commonly for serial or dial-up connections. PPPoE's main purpose is similar to that of a DHCP client/server scenario. PPPoE clients receive IP address information from an ISP acting as a PPPoE server. The advantage of PPPoE over DHCP is that it can require a username and password authentication before giving out connection information. Typical areas where this might be used are cable modems or DSL line configuration.

Configuring PPPoE on the PIX

The PIX firewall can support client PPPoE configurations only on the outside interface. To configure PPPoE, the `vpdn` command is needed. This command is a versatile command that is also used for creating VPN tunnels into the PIX. The steps to creating a PPPoE client configuration are as follows:

1. Define a VPDN group.
2. Define the VPDN group authentication.
3. Set the VPDN group ISP username.
4. Configure a VPDN username and password.
5. Enable PPPoE on the outside interface.

The vpdn group Command

The `vpdn group` command creates a group with which all parameters for the PPPoE connection will be associated. The following displays the syntax of the `vpdn group` command for PPPoE:

```
pixfirewall(config)# vpdn group <group_name> request dialout pppoe
```

Table 8.7 displays the command options for the `vpdn` command for PPPoE.

Table 8.7 vpdn group command options

Option	Function
group_name	This is the unique name you want to use for all the parameters you will send for the VPDN connection.
request dialout pppoe	This specifies that the group will be using a PPPoE connection for dial-out capabilities.

The following command demonstrates configuring a VPDN group named `ExamCram` that is using PPPoE as the requested dial-out connection:

```
pixfirewall(config)# vpdn group ExamCram request dialout pppoe
```

The vpdn group authentication Command

Just like PPP, PPPoE can use authentication. The PIX currently supports three types of authentication: PAP, CHAP, and MSCHAP. Its command syntax is shown here:

```
pixfirewall(config)# vpdn group <group_name>
              ppp authentication <pap¦chap¦mschap>
```

This command demonstrates setting a VPDN group named `ExamCram` to use PAP for authentication:

```
pixfirewall(config)# vpdn group ExamCram ppp authentication pap
```

The vpdn group localname Command

When connecting to an ISP, a username is given to the account, and this username must be linked to the VPDN group you are using for the PPPoE connection. The `localname` command links the username to the VPDN group. The command syntax is

```
pixfirewall(config)# vpdn group <group_name> localname <username>
```

The command shown here demonstrates setting a VPDN group named `ExamCram` with a local name of `danny` that will be sent to the ISP during the authentication phase:

```
pixfirewall(config)# vpdn group ExamCram localname danny
```

The vpdn username and password Command

The `vpdn group localname` command specifies only the username needed to connect to the ISP. However, the ISP also needs a password. This password is created separately from the `vpdn group` commands, but it is associated back to the group by using the same name as in the `vpdn group localname` command. For example, if you created a `localname` called `danny`, you would also create a username and password entry with `danny`. Here is the command syntax:

```
pixfirewall(config)# vpdn username <name> password <pwd>
```

This command demonstrates setting a VPDN username and password that will be sent to the ISP during the authentication phase. After it's configured, the PIX firewall will not require user interaction during the connection phase:

```
pixfirewall(config)# vpdn username danny password 123
```

After the username and password are configured for PPPoE, no user interaction is needed when the PIX acquires the IP address information from the ISP.

The ip address Command

The last step is to enable PPPoE on the outside interface. The `ip address` command is used to enable PPPoE on the interface, and its command syntax is as follows:

```
pixfirewall(config)# ip address <if_name> <ip_address> <mask>
               pppoe [setroute]
```

The `setroute` option enables you to receive the default route from the ISP PPPoE server. The following is an example of setting the outside interface to use PPPoE with the `setroute` option:

```
pixfirewall(config)# ip address outside pppoe setroute
```

A PPPoE Example

Listing 8.4 displays the five commands needed to create a PPPoE client configuration on the PIX firewall.

Listing 8.4 Example Using PPPoE

```
pixfirewall(config)# vpdn group ExamCram request dialout pppoe
pixfirewall(config)# vpdn group ExamCram ppp authentication pap
pixfirewall(config)# vpdn group ExamCram localname danny
pixfirewall(config)# vpdn username danny password 123
pixfirewall(config)# ip address outside pppoe setroute
```

Routing

The PIX firewall supports only two methods of routing—static and passive RIP. *Static* routing is the process of manually configuring a route, whereas *passive RIP* is the process of dynamically learning routes via the Routing Information Protocol (RIP) from other RIP-enabled routers. The PIX firewall does not share its routing information with other routers; it only passively listens to RIP advertisements.

After you assign an IP address to an interface, the PIX firewall creates a directly connected entry in the routing table. But any routes not directly connected need to be configured. Listing 8.5 uses the `show route` command to display the directly connected routes shown in Figure 8.5.

Listing 8.5 Connected Routes

```
pixfirewall(config)# show route
        outside 169.254.0.0 255.255.0.0 169.254.8.1 1 CONNECT static
        inside 192.168.1.0 255.255.255.0 192.168.1.1 1 CONNECT static
pixfirewall(config)#
```

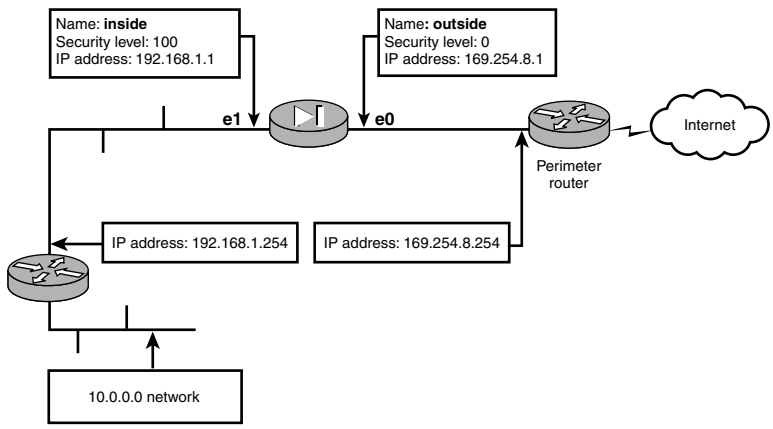

Figure 8.5 PIX network.

Static Routes

Manually configuring static routes enables the PIX firewall to direct traffic out the appropriate interface and off to the next hop. The `route` command is used to create a manual static route; its command syntax is shown here:

```
pixfirewall(config)# [no] route <if_name> <foreign_ip> <mask>
                <gateway> [<metric>]
```

Table 8.8 displays the command options for the `route` command.

Table 8.8 route Command Options

Option	Function
if_name	This is the interface name where the route exists.
foreign_ip	This is the network address to be routed. Use 0.0.0.0 for the default route.
mask	This specifies a mask to use with the **foreign_ip** option.
gateway	This is the next hop IP address to get to the network defined in the **foreign_ip** option.
metric	This specifies the hops to the network.

In Listing 8.6, two static routes are created. The first is a default route to the Internet, and the second is a static route to the 10.0.0.0 network.

Listing 8.6 Static Routes

```
pixfirewall(config)# clear route
pixfirewall(config)# route outside 0.0.0.0 0.0.0.0 169.254.8.254
pixfirewall(config)# route inside 10.0.0.0 255.0.0.0 192.168.1.254
pixfirewall(config)# show route
        outside 0.0.0.0 0.0.0.0 169.254.8.254 1 OTHER static
        inside 10.0.0.0 255.0.0.0 192.168.1.254 1 OTHER static
        outside 169.254.0.0 255.255.0.0 169.254.8.1 1 CONNECT static
        inside 192.168.1.0 255.255.255.0 192.168.1.1 1 CONNECT static
pixfirewall(config)#
```

In Listing 8.6, the first line clears all the existing routes, and the second line displays the `route` command needed to configure a static default route to the Internet according to Figure 8.5. The third line configures a static route to the 10.0.0.0 network going through the gateway of 192.168.1.254.

The Routing Information Protocol

The PIX firewall can learn routes dynamically using the routing protocols RIP v1 or RIP v2. The routing protocol RIP advertises the routes a device knows to other RIP-enabled devices. Although the PIX supports RIP, it listens to RIP advertisements only in a passive configuration. This enables the PIX to learn routes for other devices without advertising them to others. The exception to this is that the PIX can advertise a default route to another device, but it won't advertise any learned routes. The following is the `rip` command's syntax:

```
pixfirewall(config)# [no] rip <if_name> default¦passive [version <1¦2>]
               [authentication <text ¦md5> <key> <key id>]
```

Table 8.9 displays the `rip` command's options.

Table 8.9 rip Command Options

Option	Function
if_name	This is the interface name to perform RIP.
default	This broadcasts the default route on the interface.
passive	This enables passive RIP, which allows the PIX to learn RIP routes.
version 1|2	This enables version 1 or 2 RIP.
authentication	This works with RIP v2 to provide secure routing updates.

Here is an example of the using the `rip` command:

```
pixfirewall(config)# rip inside passive version 1
```

Table 8.10 lists four other helpful routing and RIP commands.

Table 8.10 General Routing Commands

Command	Function
show route	Displays a routing table
clear route	Clears a single route or the whole routing table
show rip	Displays only RIP-learned routes
debug rip	Used to display RIP traffic

To create a default route, you use the route outside **0.0.0.0 0.0.0.0 <gateway ip address>** command. This command can also be written as **route outside 0 0 <gateway ip address>**.