



Lab 9.1.9 Configure ACLs in the PIX Security Appliance using CLI

Objective

In this lab exercise, the students will complete the following tasks:

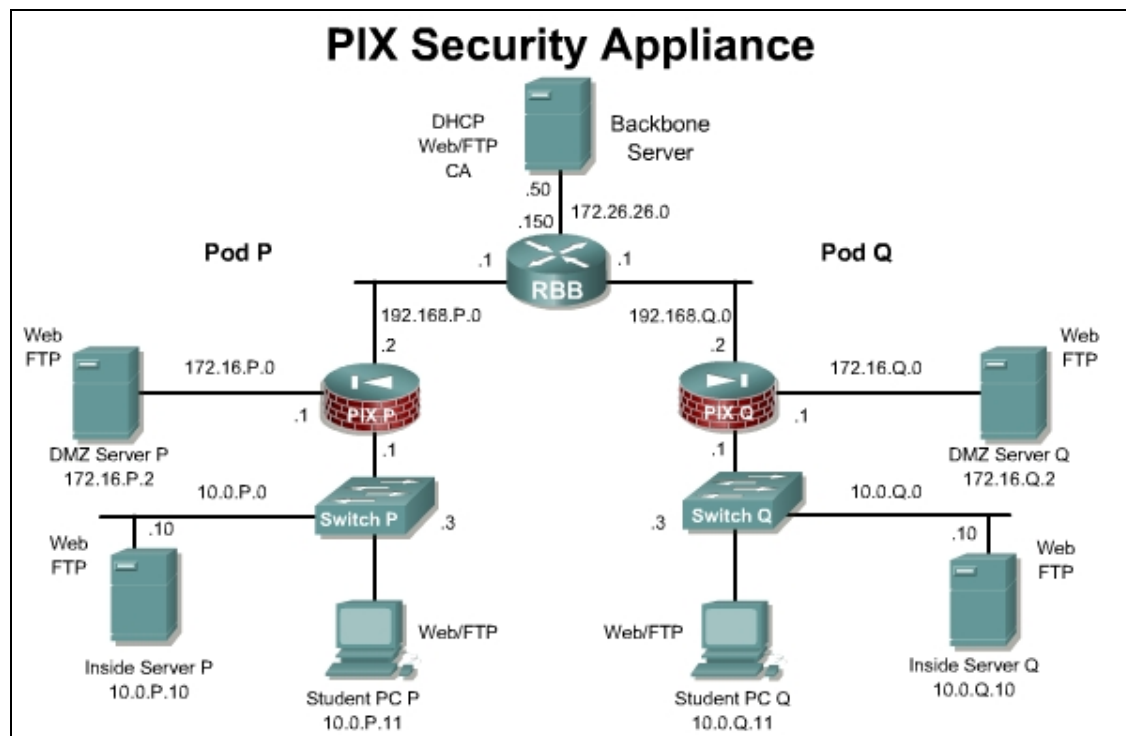
- Disable ping to an interface.
- Configure inbound and outbound access control lists (ACLs).
- Configure malicious active code filtering.

Scenario

Company XYZ has purchase and installed a PIX Security Appliance on the network. By default, the PIX does not allow any traffic from a lower security interface to a higher security interface. In order for hosts on a higher security interface to be accessed from a lower security interface, access control lists must be configured on the PIX.

Topology

This figure illustrates the lab network environment.



Preparation

Begin with the standard lab topology and verify the starting configuration on the pod PIX Security Appliance. Access the PIX Security Appliance console port using the terminal emulator on the student PC. If desired, save the PIX Security Appliance configuration to a text file for later analysis.

Tools and resources

In order to complete the lab, the following is required:

- Standard PIX Security Appliance lab topology
- Console cable
- HyperTerminal

Additional materials

Further information about the objectives covered in this lab can be found at http://www.cisco.com/en/US/products/sw/secursw/ps2120/products_installation_and_configuration_guides_list.html.

Command list

In this lab exercise, the following commands will be used. Refer to this list if assistance or help is needed during the lab exercise.

Command	Description
<code>access-list id [line line-number] [extended] {deny permit} {protocol / object-group protocol_obj_grp_id} {host source-ip / source-ip mask / interface ifc_name / object-group network_obj_grp_id / any} {host destination-ip / destination-ip mask / interface ifc_name / object-group network_obj_grp_id / any} [log [[level] [interval secs] disable default]] [inactive / time-range time_range_name]</code>	Command used to configure an access list.
<code>clear configure icmp</code>	Removes <code>icmp</code> command statements from the configuration.
<code>filteractivex {[port[-port] except } local_ip local_mask foreign_ip foreign_mask]</code>	Block outbound ActiveX, Java applets, and other HTML <object> tags from outbound packets.
<code>filter java {[port[-port] except } local_ip local_mask foreign_ip foreign_mask]</code>	Specifies to filter out Java applets returning from an outbound connection.
<code>icmp {permit deny} ip_address net_mask [icmp_type] if_name</code>	Enables or disables the ability to ping a PIX Security Appliance interface.
<code>show running-config access-list</code>	Displays the configured access lists.

Command	Description
show running-config filter	Displays URL, Java, and ActiveX filtering configurations.
url-server [(if_name)] vendor websense host local_ip [timeout seconds] [protocol {TCP UDP connections num_conns} version]	Command used to define Websense filtering.

Step 1 Disable Pingging to an Interface

Perform the following lab steps to configure an ICMP ACL to prevent pingging to the PIX Security Appliance interfaces:

- a. Ping the inside interface of the PIX Security Appliance from the inside host:

```
C:\>ping 10.0.P.1

Pinging 10.0.P.1 with 32 bytes of data:

Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128

(where P = pod number)
```

- b. Ping the outside interface from the inside host. By default, pingging through the PIX Security Appliance to a PIX Security Appliance interface is not allowed:

```
C:\>ping 192.168.P.2

Pinging 192.168.P.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

(where P = pod number)
```

- c. Use the **icmp** command to prevent pingging the inside interface:

```
PixP(config)# icmp deny any echo inside
```

1. Why would this command be used in a production network?
-

- d. View the ICMP ACL:

```
PixP(config)# show running-config icmp

icmp deny any echo inside
```

- e. Ping the inside PIX Security Appliance interface from the inside host. The ICMP ACL causes the ping to fail:

```
C:\>ping 10.0.P.1

Pinging 10.0.P.1 with 32 bytes of data:

Request timed out.
Request timed out.
```

Request timed out.

Request timed out.

(where P = pod number)

- f. Enable ping to the PIX Security Appliance inside interface:

```
PixP(config)# clear configure icmp
```

- g. Verify that the ICMP ACL is removed by pinging the inside interface of the PIX Security Appliance:

```
C:\>ping 10.0.P.1
```

Pinging 10.0.P.1 with 32 bytes of data:

```
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
```

```
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
```

```
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
```

```
Reply from 10.0.P.1: bytes=32 time<10ms TTL=128
```

(where P = pod number)

Step 2 Configure Inbound ACLs

Perform the following steps to configure ACLs:

- a. Configure the following statics for the pod bastion host and the pod inside host:

```
pixP(config)# static (dmz,outside) 192.168.P.11 bastionhost netmask  
255.255.255.255
```

```
pixP(config)# static (inside,outside) 192.168.P.10 insidehost  
netmask 255.255.255.255
```

(where P = pod number)

- b. Test web access to the bastion host of the peer pod. The peer bastion host should not be accessible by HTTP at this point.

- i. Open a web browser on the student PC.

- ii. Use the web browser to access the bastion host of the peer pod by entering:

http://192.168.Q.11.

(where Q = peer pod number)

- c. Test FTP access to the bastion host of peer pod. The peer bastion host should not be accessible by FTP at this point. Attempt to access the bastion host of another pod group using FTP:

Start > Run > ftp 192.168.Q.11

(where Q = peer pod number)

- d. Create an ACL to permit inbound HTTP and FTP access to the bastion host from the peer outside network:

```
pixP(config)# access-list ACLIN permit tcp 192.168.Q.0 255.255.255.0  
host 192.168.P.11 eq www
```

```
pixP(config)# access-list ACLIN permit tcp host 192.168.Q.10 host  
192.168.P.11 eq ftp
```

(where P = pod number, Q = peer pod number)

1. What command would be used to allow access to a mail server running on the bastion host?

- e. Add commands to permit inbound web traffic to the inside host, permit inbound pings, permit icmp echo replies to the inside host, and deny all other traffic from the Internet:

```
pixP(config)# access-list ACLIN permit tcp any host 192.168.P.10 eq www
```

```
pixP(config)# access-list ACLIN permit icmp any any echo
```

```
pixP(config)# access-list ACLIN permit icmp any host 192.168.P.10 echo-reply
```

```
pixP(config)# access-list ACLIN deny ip any any
```

(where P = pod number)

- f. Bind the ACL to the outside interface:

```
pixP(config)# access-group ACLIN in interface outside
```

- g. Create an access-list to allow icmp echo-replies from the bastion host:

```
pixP(config)# access-list ICMPDMZ permit icmp host bastionhost any echo-reply
```

- h. Bind the new ACL to the dmz interface:

```
pixP(config)# access-group ICMPDMZ in interface dmz
```

- i. Display the access-list configuration. Use the **show running-config access-list** command to display the configuration only, with no line numbers or hit counts

```
pixP(config)# show running-config access-list
```

```
access-list ACLIN extended permit tcp 192.168.Q.0 255.255.255.0 host 192.168.P.11 eq www
```

```
access-list ACLIN extended permit tcp host 192.168.Q.10 host 192.168.P.11 eq ftp
```

```
access-list ACLIN extended permit tcp any host 192.168.P.10 eq www
```

```
access-list ACLIN extended permit icmp any any echo
```

```
access-list ACLIN extended permit icmp any host 192.168.P.10 echo-reply
```

```
access-list ACLIN extended deny ip any any
```

```
access-list icmpdmz extended permit icmp host bastionhost any echo-reply
```

(where P = pod number, Q = peer pod number)

- j. Use the **show access-list** command to display the access list and observe the hit counts and line numbers:

```
pixP(config)# show access-list
```

```
access-list cached ACL log flows: total 0, denied 0 (denyflow- max 4096) alert-interval 300
```

```
access-list ACLIN; 6 elements
```

```
access-list ACLIN line 1 extended permit tcp 192.168.Q.0 255.255.255.0 host 192.168.P.11 eq www (hitcnt=0)
```

```
access-list ACLIN line 2 extended permit tcp host 192.168.Q.10 host 192.168.P.11 eq ftp
```

```
(hitcnt=0)
access-list ACLIN line 3 extended permit tcp any host 192.168.P.10
eq www (hitcnt=0)
access-list ACLIN line 4 extended permit icmp any any echo
(hitcnt=0)
access-list ACLIN line 5 extended permit icmp any host
192.168.6.10 echo-reply (hitcnt=0)
access-list ACLIN line 6 extended deny ip any any (hitcnt=0)
access-list icmpdmz; 1 elements
access-list icmpdmz line 1 extended permit icmp host bastionhost any
echo-reply (hitcnt=0)
(where P = pod number, Q = peer pod number)
```

Step 3 Test and Verify the Inbound ACLs

Perform the following steps to test the inbound ACL:

- a. Have a peer inside host ping the inside host:

```
C:\>ping 192.168.Q.10
Pinging 192.168.Q.10 with 32 bytes of data:
Reply from 192.168.Q.10: bytes=32 time<10ms TTL=128
Reply from 192.168.Q.10: bytes=32 time<10ms TTL=128
Reply from 192.168.Q.10: bytes=32 time<10ms TTL=128
Reply from 192.168.Q.10: bytes=32 time<10ms TTL=128
(where Q = peer pod number)
```

- b. Have a peer inside host ping the bastion host:

```
C:\>ping 192.168.Q.11
Pinging 192.168.Q.11 with 32 bytes of data:
Reply from 192.168.Q.11: bytes=32 time<10ms TTL=128
Reply from 192.168.Q.11: bytes=32 time<10ms TTL=128
Reply from 192.168.Q.11: bytes=32 time<10ms TTL=128
Reply from 192.168.Q.11: bytes=32 time<10ms TTL=128
(where Q = peer pod number)
```

- c. Ping the bastion host from the student PC:

```
C:\>ping 172.16.P.2
Pinging 172.16.P.2 with 32 bytes of data:
Reply from 172.16.P.2: bytes=32 time<10ms TTL=128
Reply from 172.16.P.2: bytes=32 time<10ms TTL=128
Reply from 172.16.P.2: bytes=32 time<10ms TTL=128
Reply from 172.16.P.2: bytes=32 time<10ms TTL=128
(where P = pod number)
```

- d. Ping the Backbone server from the student PC:

```
C:\>ping 172.26.26.50

Pinging 172.26.26.50 with 32 bytes of data:
Reply from 172.26.26.50: bytes=32 time<10ms TTL=128
Reply from 172.26.26.50: bytes=32 time<10ms TTL=128
Reply from 172.26.26.50: bytes=32 time<10ms TTL=128
Reply from 172.26.26.50: bytes=32 time<10ms TTL=128
```

- e. Test web access to the bastion hosts of peer pod groups by completing the following substeps. The web request should be successful when accessing the peer bastion host via its static mapping:
- Open a web browser on the student PC.
 - Use the web browser to access the bastion host of the peer pod group by entering:
http://192.168.Q.11.
(where Q = peer pod number)
 - Have a peer pod group attempt to access the bastion host in the same way.
- f. Test web access to the inside hosts of peer pod groups by completing the following substeps. Access to the IP address of the static mapped to the inside host of the opposite pod group should be successful:
- Open a web browser on the client PC.
 - Use the web browser to access the inside host of the peer pod group by entering:
http://192.168.Q.10.
(where Q = peer pod number)
 - Have a peer pod group attempt to access the inside host in the same way.
- g. Test FTP access to the bastion hosts of peer pod groups by completing the following substeps. Access to the peer bastion host via FTP should be successful:
- Using FTP, attempt to access the bastion host of a peer pod group:
Start > Run > ftp 192.168.Q.11.
(where Q = peer pod number)
 - Have a peer pod group use FTP to attempt to access their peer bastion host.
 - Were any of the above steps unsuccessful? Why?

- h. Display the access lists again and observe the hit counts:

```
PixP(config)# show access-list

access-list cached ACL log flows: total 0, denied 0 (deny-flow-max
4096)          alert-interval 300

access-list ACLIN; 6 elements

access-list ACLIN line 1 extended permit tcp 192.168.Q.0
255.255.255.0 host 192.168.P.11 eq www(hitcnt=2)

access-list ACLIN line 2 extended permit tcp host 192.168.Q.10 host
192.168.P.11 eq ftp (hitcnt=0)

access-list ACLIN line 3 extended permit tcp any host 192.168.P.10
eq www (hitcnt=2)
```

```

access-list ACLIN line 4 extended permit icmp any any echo
(hitcnt=20)

access-list ACLIN line 5 extended permit icmp any host 192.168.P.10
echo-reply (hitcnt=12)

access-list ACLIN line 6 extended deny ip any any (hitcnt=0)

access-list ICMPDMZ; 1 elements

access-list ICMPDMZ line 1 extended permit icmp host bastionhost any
echo-reply (hitcnt=12)

```

(where P = pod number, Q = peer pod number)

Step 4 Configure an Outbound ACL

Perform the following lab steps to configure ACLs:

- Deny outbound web traffic.
 - Allow outbound FTP traffic from the internal network to 172.26.26.50.
- a. Test web access to the Internet by completing the following substeps. The test to access 172.26.26.50 should be successful:
 - i. Open a web browser on the student PC.
 - ii. Use the web browser to access Internet host 172.26.26.50 by entering:
http://172.26.26.50.
 - b. Test FTP access to Internet host 172.26.26.50. Access to the host 172.26.26.50 via FTP should be successful:

On the FTP client, attempt to access host 172.26.26.50:

Start>Run>ftp 172.26.26.50

- c. Create an ACL that prevents users on the internal network from making outbound HTTP connections:

```
PixP(config)# access-list ACLOUT deny tcp any any eq www
```

This access list prevents all outbound connections.

- d. Enter the **access-group** command to create an access group that will bind the ACL to an interface:

```
PixP(config)# access-group ACLOUT in interface inside
```

- e. Display the configured access lists, and observe the hit count:

```

PixP(config)# show access-list

access-list ACLIN; 6 elements

access-list ACLIN line 1 extended permit tcp 192.168.Q.0
255.255.255.0 host 192.168.P.11 eq www(hitcnt=4)

access-list ACLIN line 2 extended permit tcp host 192.168.Q.10 host
192.168.P.11 eq ftp (hitcnt=1)

access-list ACLIN line 3 extended permit tcp any host 192.168.P.10
eq www (hitcnt=4)

access-list ACLIN line 3 extended permit icmp any any echo
(hitcnt=20)

access-list ACLIN line 5 extended permit icmp any host 192.168.P.10
echo-reply (hitcnt=12)

access-list ACLIN line 6 extended deny ip any any (hitcnt=0)

```



```

access-list ICMPDMZ; 1 elements
access-list ICMPDMZ line 1 extended permit icmp host bastionhost any
echo-reply (hitcnt=12)
access-list ACLOUT; 1 elements
access-list ACLOUT line 1 extended deny tcp any any eq www
(hitcnt=0)

```

(where P = pod number, Q = peer pod number)

- f. Test web access to the Internet by completing the following substeps. The test via HTTP should fail.

- i. Open a web browser on the student PC.
- ii. Use the web browser to access the Internet by entering:

http://172.26.26.50.

- g. Test FTP access to an Internet host. The FTP connection should fail as well due to the implicit deny any:

On the FTP client, attempt to access host 172.26.26.50:

Start>Run>ftp 172.26.26.50

- h. Display the access list again and note that the hit count has incremented:

```

PixP(config)# show access-list
access-list ACLIN; 6 elements
access-list ACLIN line 1 extended permit tcp 192.168.Q.0
255.255.255.0 host 192.168.P.11 eq www (hitcnt=4)
access-list ACLIN line 2 extended permit tcp host 192.168.Q.10 host
192.168.P.11 eq ftp (hitcnt=1)
access-list ACLIN line 3 extended permit tcp any host 192.168.P.10
eq www (hitcnt=4)
access-list ACLIN line 4 extended permit icmp any any echo
(hitcnt=20)
access-list ACLIN line 5 extended permit icmp any host 192.168.P.10
echo-reply (hitcnt=12)
access-list ACLIN line 6 extended deny ip any any (hitcnt=0)
access-list ICMPDMZ; 1 elements
access-list ICMPDMZ line 1 extended permit icmp host bastionhost any
echo-reply (hitcnt=12)
access-list ACLOUT; 1 elements
access-list ACLOUT line 1 extended deny tcp any any eq www
(hitcnt=3)

```

(where P = pod number, Q = peer pod number)

- i. Add an additional command to the ACL to permit outbound FTP access to host 172.26.26.50:

```

PixP(config)# access-list ACLOUT permit tcp 10.0.P.0 255.255.255.0
host 172.26.26.50 eq ftp

```

(where P = pod number)

- j. Add another access list command statement to deny other outbound IP traffic:

```

PixP(config)# access-list ACLOUT deny ip any any

```

This access list statement is only needed to enable viewing of the hit counts.

- k. View the access list again:

```
PixP(config)# show access-list ACLOUT
access-list ACLOUT; 3 elements
access-list ACLOUT line 1 extended deny tcp any any eq www
(hitcnt=3)
access-list ACLOUT line 2 extended permit tcp 10.0.P.0 255.255.255.0
host 172.26.26.50 eq ftp (hitcnt=0)
access-list ACLOUT line 3 extended deny ip any any (hitcnt=0)
(where P = pod number)
```

Step 5 Test and Verify the Outbound ACL

Perform the following steps to test the outbound ACL:

- a. Test web access to the Internet by completing the following substeps. Access to the Internet host will fail due to the deny ACL:
- Open a web browser on the student PC.
 - Use the web browser to attempt to access the Internet by entering:
http://172.26.26.50.
- b. Test FTP access to an Internet host by performing the following on the FTP client. At this point, a connection using FTP will work:

Start>Run>ftp 172.26.26.50

- c. Test the FTP access to a peer pod bastion host by attempting to access the peer pod bastion host on the FTP client. The connection using FTP should fail:

Start>Run>ftp 192.168.Q.11

(where Q = peer pod number)

- d. View the outbound access list again and observe the hit counts:

```
PixP(config)# show access-list ACLOUT
access-list ACLOUT line 1 extended deny tcp any any eq www
(hitcnt=2)
access-list ACLOUT line 2 extended permit tcp 10.0.P.0 255.255.255.0
host 172.26.26.50 eq ftp (hitcnt=1)
access-list ACLOUT line 3 extended deny ip any any (hitcnt=3)
```

(where P = pod number)

Be sure to enter the following command exactly as shown. If the ACL name is omitted all access list statements are removed.

- e. Remove the outbound ACL:

```
PixP(config)# clear configure access-list ACLOUT
```

- f. Verify that the outbound ACL has been removed:

```
PixP(config)# show access-list
access-list ACLIN; 6 elements
access-list ACLIN permit tcp 192.168.Q.0 255.255.255.0 host
192.168.P.11 eq www(hitcnt=4)
```

```

access-list ACLIN permit tcp host 192.168.Q.10 host 192.168.P.11 eq
ftp (hitcnt= 1)
access-list ACLIN permit tcp any host 192.168.P.10 eq www (hitcnt=4)
access-list ACLIN permit icmp any any echo (hitcnt=20)
access-list ACLIN permit icmp any host 192.168.P.10 echo-reply
(hitcnt=12)
access-list ACLIN deny ip any any (hitcnt=0)
access-list ICMPDMZ; 1 elements
access-list ICMPDMZ permit icmp host bastionhost any echo-reply
(hitcnt=12)

```

(where P = pod number, Q = peer pod number)

- g. View the access groups:

```

PixP(config)# show running-config access-group
access-group ACLIN in interface outside
access-group ICMPDMZ in interface dmz
pixP(config)#

```

Save the configuration:

```

PixP(config)# write memory

```

Step 6 Filter Malicious Active Code

Perform the following lab steps to configure ActiveX and filter Java.

Note If the ActiveX and Java applets are not working properly, the security settings in the web browser may need to be adjusted to allow these applets to run. The Java Virtual Machine must be running for the Java Applet to run. Also, any popup blockers that are running on the student PCs must be disabled, as the links for the applets on the pod homepage will launch the applets in a new window.

- a. Enter **http://192.168.Q.10** in the web browser. After the peer pods homepage appears, click on the **ActiveX Control** link. The ActiveX Control should open successfully.

Did the ActiveX Control open successfully?

- b. On the PIX Security Appliance, enter the **filter activex** command to block ActiveX from any local host and for connections to any foreign host on port 80:

```

PixP(config)# filter activex 80 0 0 0 0

```

- a. What is the significance of 0 0 0 0?
-

- c. Open a new web browser and enter **http:192.168.Q.10**. After the webpage opens, click on the **ActiveX Control** link. The ActiveX Control should not open successfully.

Note: It might be necessary to clear the web browser cache. In Internet Explorer, go to **Tools > Internet Options....** and click the **Delete Files** button in the **Temporary Internet files** area.

Did the ActiveX Control open successfully?

- d. Enter **http://192.168.Q.10** in the web browser. After the peer pods homepage appears, click on the **Java Applet** link. The Java Applet should open successfully.

Did the Java Applet open successfully?

- e. Enter the **filter java** command to block Java applets:

```
PixP(config)# filter java 80 0 0 0 0
```

- f. Open a new web browser and enter **http:192.168.Q.10**. After the webpage opens, click on the **Java Applet** link. The Java Applet should not open successfully.

Note: It might be necessary to clear the web browser cache.

Did the Java Applet open successfully?

- g. Use the following command to show the filters:

```
PixP(config)# show running-config filter
filter activex 80 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
filter java 80 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
```

Step 7 Configure the PIX Security Appliance to Work with a URL Filtering Server

Perform the following steps to configure the PIX Security Appliance to work with a URL-filtering server:

- a. Enter the **url-server** command to designate the URL-filtering server:

```
PixP(config)# url-server (inside) host 10.0.P.11 timeout 30 protocol
TCP version 4
```

- b. Show the designated url-server by entering the following command:

```
PixP(config)# show running-config url-server
url-server (inside) vendor websense host insidehost timeout 30
protocol TCP version 4 connections 5
```

- c. Enter the **filter url http** command to prevent outbound users from accessing WWW URLs that are designated with the filtering application:

```
PixP(config)# filter url http 0 0 0 0 allow
```

- d. Display the **filter url http** command by using the following command:

```
PixP(config)# show running-config filter
filter activex 80 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
filter java 80 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
filter url http 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 allow
```

- e. Remove the **filter** commands from the configuration:

```
PixP(config)# clear configure filter
```

- f. Remove the **url-server** command:

```
PixP(config)# no url-server (inside) host insidehost
(where P = the pod number)
```

- g. Save the configuration:

```
PixP(config)# write memory
```

Step 8 Download, Install, and Configure a URL Filtering Server (OPTIONAL)

If time permits, download, install, and configure a web filtering server. A Cisco IOS Firewall is also able to interoperate with Websense and N2H2 servers to provide web filtering.

Websense

<http://www.websense.com/downloads/>

http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_tech_note09186a00801e4197.shtml

N2H2

<http://www.n2h2.com/products/bess.php?device=pix>