CHAPTER 39

Applying Filtering Services

This chapter describes how filtering can provide greater control over traffic passing through the ASA. Filtering can be used in two distinct ways:

- Filtering ActiveX objects or Java applets
- Filtering with an external filtering server

Instead of blocking access altogether, you can remove specific undesirable objects from HTTP traffic, such as ActiveX objects or Java applets, that may pose a security threat in certain situations.

You can also use URL filtering to direct specific traffic to an external filtering server, such an Secure Computing SmartFilter (formerly N2H2) or Websense filtering server. Long URL, HTTPS, and FTP filtering can now be enabled using both Websense and Secure Computing SmartFilter for URL filtering. Filtering servers can block traffic to specific sites or types of sites, as specified by the security policy.

Note URL caching will only work if the version of the URL server software from the URL server vender supports it.

Because URL filtering is CPU-intensive, using an external filtering server ensures that the throughput of other traffic is not affected. However, depending on the speed of your network and the capacity of your URL filtering server, the time required for the initial connection may be noticeably slower when filtering traffic with an external filtering server.

This chapter includes the following sections:

Configuring ActiveX Filtering

This section includes the following topics:

Information About ActiveX Filtering

ActiveX objects may pose security risks because they can contain code intended to attack hosts and servers on a protected network. You can disable ActiveX objects with ActiveX filtering.

ActiveX controls, formerly known as OLE or OCX controls, are components you can insert in a web page or other application. These controls include custom forms, calendars, or any of the extensive third-party forms for gathering or displaying information. As a technology, ActiveX creates many potential problems for network clients including causing workstations to fail, introducing network security problems, or being used to attack servers.

The **filter activex** command blocks the HTML <object> commands by commenting them out within the HTML web page. ActiveX filtering of HTML files is performed by selectively replacing the <APPLET> and </APPLET> and <OBJECT CLASSID> and </OBJECT> tags with comments. Filtering of nested tags is supported by converting top-level tags to comments.

Caution This command also blocks any Java applets, image files, or multimedia objects that are embedded in object tags.

If the <object> or </object> HTML tags split across network packets or if the code in the tags is longer than the number of bytes in the MTU, ASA cannot block the tag.

ActiveX blocking does not occur when users access an IP address referenced by the **alias** command or for WebVPN traffic.

Licensing Requirements for ActiveX Filtering

The following table shows the licensing requirements for this feature:

Table 39-1 Licensing Requirements

Model	License Requirement
All models	Base License.

Configuring ActiveX Filtering

To remove ActiveX objects in HTTP traffic passing through the ASA, enter the following command:

Command	Purpose
filter activex *port*[**-***port*] *local_ip local_mask foreign_ip foreign_mask*	Removes ActiveX objects.

To use this command, replace port with the TCP port to which filtering is applied. Typically, this is port 80, but other values are accepted. The **http** or **url** literal can be used for port 80. You can specify a range of ports by using a hyphen between the starting port number and the ending port number.

The local IP address and mask identify one or more internal hosts that are the source of the traffic to be filtered. The foreign address and mask specify the external destination of the traffic to be filtered.

Configuration Examples for ActiveX Filtering

You can set either address to **0.0.0.0** (or in shortened form, **0**) to specify all hosts. You can use **0.0.0.0** for either mask (or in shortened form, **0**) to specify all hosts. This command specifies that the ActiveX object blocking applies to web traffic on port 80 from any local host and for connections to any foreign host.

The following example shows how to configure activeX filtering to block all outbound connections. To remove the configuration, use the **no** form of the command:

```
hostname(config)# filter activex 80 0 0 0 0
hostname(config)# no filter activex 80 0 0 0 0
```

Feature History for ActiveX Filtering

Table 39-2 lists the release history for ActiveX Filtering.

Table 39-2 Feature History for ActiveX Filtering

Feature Name	Releases	Feature Information
Filter activex	7.0	This command was preexisting.

Configuring Java Applet Filtering

This section includes the following topics:

Information About Java Applet Filtering

The **filter java** command filters out Java applets that return to the ASA from an outbound connection. The user still receives the HTML page, but the web page source for the applet is commented out so that the applet cannot execute. The **filter java** command does not filter WebVPN traffic.

Java applets may pose security risks because they can contain code intended to attack hosts and servers on a protected network. You can remove Java applets with the **filter java** command.

Note Use the **filter activex** command to remove Java applets that are embedded in <object> tags.

Licensing Requirements for Java Applet Filtering

The following table shows the licensing requirements for java applet filtering:

Table 39-3 Licensing Requirements

Model	License Requirement
All models	Base License.

Configuring Java Applet Filtering

To apply filtering to remove Java applets from HTTP traffic passing through the ASA, enter the following command:

Command	Purpose
filter java *port*[**-***port*] *local_ip local_mask foreign_ip foreign_mask*	Removes Java applets in HTTP traffic passing through the ASA.

To use this command, replace port with the TCP port to which filtering is applied. Typically, this is port 80, but other values are accepted. The **http** or **url** literal can be used for port 80. You can specify a range of ports by using a hyphen between the starting port number and the ending port number.

The local IP address and mask identify one or more internal hosts that are the source of the traffic to be filtered. The foreign address and mask specify the external destination of the traffic to be filtered.

You can set either address to **0.0.0.0** (or in shortened form, **0**) to specify all hosts. You can use **0.0.0.0** for either mask (or in shortened form, **0**) to specify all hosts.

You can set either address to **0.0.0.0** (or in shortened form, **0**) to specify all hosts. You can use **0.0.0.0** for either mask (or in shortened form, **0**) to specify all hosts.

Configuration Examples for Java Applet Filtering

The following example shows how to configure java applet filtering:

The following example specifies that Java applets are blocked on all outbound connections:

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

This command specifies that the Java applet blocking applies to web traffic on port 80 from any local host and for connections to any foreign host.

The following example blocks downloading of Java applets to a host on a protected network:

```
hostname(config)# filter java http 192.168.3.3 255.255.255.255 0 0
```

This command prevents host 192.168.3.3 from downloading Java applets.

To remove the configuration, use the **no** form of the command, as in the following example:

```
hostname(config)# no filter java http 192.168.3.3 255.255.255.255 0 0
```

Feature History for Java Applet Filtering

Table 39-2 lists the release history for java applet filtering.

Table 39-4 Feature History for Java Applet Filtering

Feature Name	Releases	Feature Information
filter java	7.0	This command was preexisting.

Configuring URLs and FTP Requests with an External Server

This section describes how to filter URLs and FTP requests with an external server. This section includes the following topics:

Information About URL Filtering

You can apply filtering to connection requests originating from a more secure network to a less secure network. Although you can use ACLs to prevent outbound access to specific content servers, managing usage this way is difficult because of the size and dynamic nature of the Internet. You can simplify configuration and improve security appliance performance by using a separate server running one of the following Internet filtering products:

- Websense Enterprise for filtering HTTP, HTTPS, and FTP.
- Secure Computing SmartFilter (formerly N2H2) for filtering HTTP, HTTPS, FTP, and long URL filtering.

Note URL caching will only work if the version of the URL server software from the URL server vender supports it.

Although security appliance performance is less affected when using an external server, users may notice longer access times to websites or FTP servers when the filtering server is remote from the security appliance.

When filtering is enabled and a request for content is directed through the security appliance, the request is sent to the content server and to the filtering server at the same time. If the filtering server allows the connection, the security appliance forwards the response from the content server to the originating client. If the filtering server denies the connection, the security appliance drops the response and sends a message or return code indicating that the connection was not successful.

If user authentication is enabled on the security appliance, then the security appliance also sends the user name to the filtering server. The filtering server can use user-specific filtering settings or provide enhanced reporting regarding usage.

Licensing Requirements for URL Filtering

The following table shows the licensing requirements for url filtering:

Table 39-5 Licensing Requirements

Model	License Requirement
All models	Base License.

Identifying the Filtering Server

You can identify up to four filtering servers per context. The ASA uses the servers in order until a server responds. You can only configure a single type of server (Websense or Secure Computing SmartFilter) in your configuration.

Note You must add the filtering server before you can configure filtering for HTTP or HTTPS with the **filter** command. If you remove the filtering servers from the configuration, then all **filter** commands are also removed.

Identify the address of the filtering server using the **url-server** command:

- For Websense:

```
hostname(config)# url-server (if_name) host local_ip [timeout seconds] [protocol TCP | UDP
version [1|4] [connections num_conns] ]
```

- For Secure Computing SmartFilter (formerly N2H2):

```
hostname(config)# url-server (if_name) vendor {secure-computing | n2h2} host
<local_ip> [port <number>] [timeout <seconds>] [protocol {TCP [connections <number>]} |
UDP]
```

where *<if_name>* is the name of the security appliance interface connected to the filtering server (the default is inside).

For the **vendor** {secure-computing | n2h2}, you can use ‘secure-computing as a vendor string, however, ‘n2h2’ is acceptable for backward compatibility. When the configuration entries are generated, ‘secure-computing’ is saved as the vendor string.

The **host** *<local_ip>* is the IP address of the URL filtering server.

The **port** *<number>* is the Secure Computing SmartFilter server port number of the filtering server; the ASA also listens for UDP replies on this port.

Note The default port is 4005. This is the default port used by the Secure Computing SmartFilter server to communicate to the ASA via TCP or UDP. For information on changing the default port, please refer to the *Filtering by N2H2 Administrator's Guide*.

The **timeout** <*seconds*> is the number of seconds the security appliance should keep trying to connect to the filtering server.

The **connections** <*number*> is the number of tries to attempt to make a connection between the host and server.

For example, to identify a single Websense filtering server, enter the following command:

```
hostname(config)# url-server (perimeter) host 10.0.1.1 protocol TCP version 4
```

This identifies a Websense filtering server with the IP address 10.0.1.1 on a perimeter interface of the ASA.Version 4, which is enabled in this example, is recommended by Websense because it supports caching.

To identify redundant Secure Computing SmartFilter servers, enter the following commands:

```
hostname(config)# url-server (perimeter) vendor n2h2 host 10.0.1.1
hostname(config)# url-server (perimeter) vendor n2h2 host 10.0.1.2
```

This identifies two Sentian filtering servers, both on a perimeter interface of the ASA.

Buffering the Content Server Response

When a user issues a request to connect to a content server, the ASA sends the request to the content server and to the filtering server at the same time. If the filtering server does not respond before the content server, the server response is dropped. This delays the web server response from the point of view of the web client because the client must reissue the request.

By enabling the HTTP response buffer, replies from web content servers are buffered and the responses are forwarded to the requesting client if the filtering server allows the connection. This prevents the delay that might otherwise occur.

To configure buffering for responses to HTTP or FTP requests, perform the following steps:

	Command	Purpose
Step 1	**url-block block** *block-buffer-limit*	Enables buffering of responses for HTTP or FTP requests that are pending a response from the filtering server. Replace *block-buffer* with the maximum number of HTTP responses that can be buffered while awaiting responses from the url-server. **Note** Buffering URLs longer than 3072 bytes are not supported.
Step 2	**url-block mempool-size** memory-pool-size	Configures the maximum memory available for buffering pending URLs (and for buffering long URLs). Replace memory-pool-size with a value from 2 to 10240 for a maximum memory allocation of 2 KB to 10 MB.

Caching Server Addresses

After a user accesses a site, the filtering server can allow the ASA to cache the server address for a certain amount of time, as long as every site hosted at the address is in a category that is permitted at all times. Then, when the user accesses the server again, or if another user accesses the server, the ASA does not need to consult the filtering server again.

Note Requests for cached IP addresses are not passed to the filtering server and are not logged. As a result, this activity does not appear in any reports. You can accumulate Websense run logs before using the **url-cache** command.

Use the **url-cache** command if needed to improve throughput, as follows:

Command	Purpose
`url-cache dst | src_dst size`	Replace *size* with a value for the cache size within the range 1 to 128 (KB). Use the **dst** keyword to cache entries based on the URL destination address. Select this mode if all users share the same URL filtering policy on the Websense server. Use the **src_dst** keyword to cache entries based on both the source address initiating the URL request as well as the URL destination address. Select this mode if users do not share the same URL filtering policy on the Websense server.

Filtering HTTP URLs

This section describes how to configure HTTP filtering with an external filtering server. This section includes the following topics:

Configuring HTTP Filtering

You must identify and enable the URL filtering server before enabling HTTP filtering.

When the filtering server approves an HTTP connection request, the ASA allows the reply from the web server to reach the originating client. If the filtering server denies the request, the ASA redirects the user to a block page, indicating that access was denied.

To enable HTTP filtering, enter the following command:

Command	Purpose
filter url [**http** | *port*[*-port*] *local_ip local_mask foreign_ip foreign_mask*] [**allow**] [**proxy-block**]	Replace *port* with one or more port numbers if a different port than the default port for HTTP (80) is used. Replace *local_ip* and *local_mask* with the IP address and subnet mask of a user or subnetwork making requests. Replace *foreign_ip* and *foreign_mask* with the IP address and subnet mask of a server or subnetwork responding to requests. The **allow** option causes the ASA to forward HTTP traffic without filtering when the primary filtering server is unavailable. Use the **proxy-block** command to drop all requests to proxy servers.

Enabling Filtering of Long HTTP URLs

By default, the ASA considers an HTTP URL to be a long URL if it is greater than 1159 characters. You can increase the maximum length allowed.

Configure the maximum size of a single URL with the following command:

Command	Purpose
url-block url-size long-url-size	Replace long-url-size with the maximum size in KB for each long URL being buffered. For Websense, this is a value from 2 to 4 for a maximum URL size of 2 KB to 4 KB; for Secure Computing, this is a value between 2 to 3 for a maximum URL size of 2 KB to 3 KB. The default value is 2.

Truncating Long HTTP URLs

By default, if a URL exceeds the maximum permitted size, then it is dropped. To avoid this, you can set the ASA to truncate a long URL by entering the following command:

Command	Purpose
filter url [**longurl-truncate** | **longurl-deny** | **cgi-truncate**]	The **longurl-truncate** option causes the ASA to send only the hostname or IP address portion of the URL for evaluation to the filtering server when the URL is longer than the maximum length permitted. Use the **longurl-deny** option to deny outbound URL traffic if the URL is longer than the maximum permitted. Use the **cgi-truncate** option to truncate CGI URLs to include only the CGI script location and the script name without any parameters. Many long HTTP requests are CGI requests. If the parameters list is very long, waiting and sending the complete CGI request including the parameter list can use up memory resources and affect ASA performance.

Exempting Traffic from Filtering

Command	Purpose
filter url except *source_ip source_mask dest_ip dest_mask*	Exempts specific traffic from filtering.

For example, the following commands cause all HTTP requests to be forwarded to the filtering server except for those from 10.0.2.54.

```
hostname(config)# filter url http 0 0 0 0
hostname(config)# filter url except 10.0.2.54 255.255.255.255 0 0
```

Filtering HTTPS URLs

You must identify and enable the URL filtering server before enabling HTTPS filtering.

Note Websense and Smartfilter currently support HTTPS; older versions of Secure Computing SmartFilter (formerly N2H2) did not support HTTPS filtering.

Because HTTPS content is encrypted, the ASA sends the URL lookup without directory and filename information. When the filtering server approves an HTTPS connection request, the ASA allows the completion of SSL connection negotiation and allows the reply from the web server to reach the originating client. If the filtering server denies the request, the ASA prevents the completion of SSL connection negotiation. The browser displays an error message such as "The Page or the content cannot be displayed."

Note The ASA does not provide an authentication prompt for HTTPS, so a user must authenticate with the ASA using HTTP or FTP before accessing HTTPS servers.

Command	Purpose
filter https *port*[*-port*] *localIP local_mask foreign_IP foreign_mask* [**allow**]	Enables HTTPS filtering. Replace *port*[*-port*] with a range of port numbers if a different port than the default port for HTTPS (443) is used. Replace *local_ip* and *local_mask* with the IP address and subnet mask of a user or subnetwork making requests. Replace *foreign_ip* and *foreign_mask* with the IP address and subnet mask of a server or subnetwork responding to requests. The **allow** option causes the ASA to forward HTTPS traffic without filtering when the primary filtering server is unavailable.

Filtering FTP Requests

You must identify and enable the URL filtering server before enabling FTP filtering.

Note Websense and Smartfilter currently support FTP; older versions of Secure Computing SmartFilter (formerly known as N2H2) did not support FTP filtering.

When the filtering server approves an FTP connection request, the ASA allows the successful FTP return code to reach originating client. For example, a successful return code is "250: CWD command successful." If the filtering server denies the request, alters the FTP return code to show that the connection was denied. For example, the ASA changes code 250 to "550 Requested file is prohibited by URL filtering policy."

Command	Purpose
filter ftp *port*[*-port*] *localIP local_mask foreign_IP foreign_mask* [**allow**] [**interact-block**]	Enables FTP filtering. Replace *port*[*-port*] with a range of port numbers if a different port than the default port for FTP (21) is used. Replace *local_ip* and *local_mask* with the IP address and subnet mask of a user or subnetwork making requests. Replace *foreign_ip* and *foreign_mask* with the IP address and subnet mask of a server or subnetwork responding to requests. The **allow** option causes the ASA to forward HTTPS traffic without filtering when the primary filtering server is unavailable.

Use the **interact-block** option to prevent interactive FTP sessions that do not provide the entire directory path. An interactive FTP client allows the user to change directories without typing the entire path. For example, the user might enter **cd ./files** instead of **cd /public/files**.

Viewing Filtering Statistics and Configuration

This section describes how to monitor filtering statistics. This section includes the following topics:

Viewing Filtering Server Statistics

To show information about the filtering server, enter the following command:

```
hostname# show url-server
```

The following is sample output from the **show url-server** command:

```
hostname# show url-server
url-server (outside) vendor n2h2 host 128.107.254.202 port 4005 timeout 5 protocol TCP
```

To show information about the filtering server or to show statistics, enter the following command:

The following is sample output from the **show url-server statistics** command, which shows filtering statistics:

```
hostname# show url-server statistics

Global Statistics:
--------------------
URLs total/allowed/denied         13/3/10
URLs allowed by cache/server      0/3
URLs denied by cache/server       0/10
HTTPSs total/allowed/denied       138/137/1
HTTPSs allowed by cache/server    0/137
HTTPSs denied by cache/server     0/1
FTPs total/allowed/denied         0/0/0
FTPs allowed by cache/server      0/0
FTPs denied by cache/server       0/0
Requests dropped                  0
Server timeouts/retries           0/0
Processed rate average 60s/300s   0/0 requests/second
Denied rate average 60s/300s      0/0 requests/second
Dropped rate average 60s/300s     0/0 requests/second

Server Statistics:
--------------------
10.125.76.20                      UP
  Vendor                          websense
  Port                            15868
  Requests total/allowed/denied   151/140/11
  Server timeouts/retries         0/0
  Responses received              151
  Response time average 60s/300s  0/0

URL Packets Sent and Received Stats:
------------------------------------
Message                 Sent    Received
STATUS_REQUEST          1609    1601
LOOKUP_REQUEST          1526    1526
LOG_REQUEST             0       NA

Errors:
-------
RFC noncompliant GET method     0
URL buffer update failure       0
```

Viewing Buffer Configuration and Statistics

The **show url-block** command displays the number of packets held in the url-block buffer and the number (if any) dropped due to exceeding the buffer limit or retransmission.

The following is sample output from the **show url-block** command:

```
hostname# show url-block
    url-block url-mempool 128
    url-block url-size 4
    url-block block 128
```

This shows the configuration of the URL block buffer.

The following is sample output from the **show url-block block statistics** command:

```
hostname# show url-block block statistics

URL Pending Packet Buffer Stats with max block  128
-----------------------------------------------------
Cumulative number of packets held:              896
Maximum number of packets held (per URL):       3
Current number of packets held (global):        38
Packets dropped due to
       exceeding url-block buffer limit:        7546
       HTTP server retransmission:              10
Number of packets released back to client:      0
```

This shows the URL block statistics.

Viewing Caching Statistics

The following is sample output from the **show url-cache stats** command:

```
hostname# show url-cache stats
URL Filter Cache Stats
----------------------
    Size :     128KB
 Entries :      1724
  In Use :       456
 Lookups :        45
    Hits :         8

This shows how the cache is used.
```

Viewing Filtering Performance Statistics

The following is sample output from the **show perfmon** command:

```
hostname# show perfmon
PERFMON STATS:    Current      Average
Xlates               0/s          0/s
Connections          0/s          2/s
TCP Conns            0/s          2/s
UDP Conns            0/s          0/s
URL Access           0/s          2/s
URL Server Req       0/s          3/s
TCP Fixup            0/s          0/s
TCPIntercept         0/s          0/s
HTTP Fixup           0/s          3/s
FTP Fixup            0/s          0/s
AAA Authen           0/s          0/s
AAA Author           0/s          0/s
AAA Account          0/s          0/s
```

This shows URL filtering performance statistics, along with other performance statistics. The filtering statistics are shown in the URL Access and URL Server Req rows.

Viewing Filtering Configuration

The following is sample output from the **show filter** command:

```
hostname# show filter
filter url http 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
```

Feature History for URL Filtering

Table 39-2 lists the release history for url filtering.

Table 39-6 Feature History for URL Filtering

Feature Name	Releases	Feature Information
filter url	7.0	This command was preexisting.