



CHAPTER 16

Applying Filtering Services

This chapter describes ways to filter web traffic to reduce security risks or prevent inappropriate use. This chapter contains the following sections:

- [Filtering Overview, page 16-1](#)
- [Filtering ActiveX Objects, page 16-1](#)
- [Filtering Java Applets, page 16-3](#)
- [Filtering URLs and FTP Requests with an External Server, page 16-4](#)
- [Viewing Filtering Statistics and Configuration, page 16-9](#)

Filtering Overview

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

- Filtering ActiveX objects or Java applets
- Filtering URLs 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 as Secure Computing SmartFilter (formerly N2H2) or Websense filtering server. Filtering servers can block traffic to specific sites or types of sites, as specified by the security policy.

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.

Filtering ActiveX Objects

This section describes how to apply filtering to remove ActiveX objects from HTTP traffic passing through the firewall. This section includes the following topics:

- [ActiveX Filtering Overview, page 16-2](#)
- [Enabling ActiveX Filtering, page 16-2](#)

ActiveX Filtering Overview

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 **filteractivex** 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, FWSM cannot block the tag.

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

Enabling ActiveX Filtering

This section describes how to remove ActiveX objects in HTTP traffic passing through the FWSM. To remove ActiveX objects, enter the following command in global configuration mode:

```
hostname(config)# filteractivex {port[-port] | except} local_ip local_mask foreign_ip foreign_mask
```

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.

To create an exception to a previous filter condition, specify the keyword **except**.



Note

The filter exception rule works only when you use the default port.

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.

The following example specifies that ActiveX objects are blocked on all outbound connections:

```
hostname(config)# filteractivex 80 0 0 0 0
```

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.

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

```
hostname(config)# no filteractivex 80 0 0 0 0
```

Filtering Java Applets

This section describes how to apply filtering to remove Java applets from HTTP traffic passing through the firewall. 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.

The **filter java** command filters out Java applets that return to the FWSM 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.



Note

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

To remove Java applets in HTTP traffic passing through the FWSM, enter the following command in global configuration mode:

```
hostname(config)# filter java {port[-port] | except} local_ip local_mask foreign_ip
foreign_mask
```

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.

To create an exception to a previous filter condition, specify the keyword **except**.



Note

The filter exception rule works only when you use the default port.

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.

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
```

Filtering 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:

- [URL Filtering Overview, page 16-4](#)
- [Identifying the Filtering Server, page 16-4](#)
- [Buffering the Content Server Response, page 16-5](#)
- [Caching Server Addresses, page 16-6](#)
- [Filtering HTTP URLs, page 16-6](#)
- [Filtering HTTPS URLs, page 16-8](#)
- [Filtering FTP Requests, page 16-9](#)

URL Filtering Overview

You can apply filtering to connection requests originating from a more secure network to a less secure network. Although you can use access lists 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 FWSM 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 and long URL filtering.

Although FWSM 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 FWSM.

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

If user authentication is enabled on the FWSM, then the FWSM also sends the username to the filtering server. The filtering server can use username filtering settings or provide enhanced reporting regarding usage.

Identifying the Filtering Server

You can identify up to four filtering servers per context. The FWSM uses the servers in order until a server responds. You can only configure a single type of server (Websense or N2H2) in your configuration.

**Note**

You must add the filtering server before you can configure filtering for HTTP or HTTPS with the **filter** command. You must also remove all filtering command before you remove the filtering servers from the configuration.

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
connections number| UDP version 1|4]
```

For Secure Computing SmartFilter (formerly N2H2):

```
hostname(config)# url-server (if_name) vendor {smartfilter | 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.

For the **vendor** {smartfilter | n2h2}, you can use 'smartfilter' as a vendor string, however, 'n2h2' is acceptable for backward compatibility. When the configuration entries are generated, 'smartfilter' 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 FWSM 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 FWSM 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 FWSM. 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 FWSM.

Buffering the Content Server Response

When a user issues a request to connect to a content server, the FWSM 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:

Step 1 To enable buffering of responses for HTTP or FTP requests that are pending a response from the filtering server, enter the following command:

```
hostname(config)# url-block block block-buffer-limit
```

Replace *block-buffer-limit* with the maximum number of blocks that will be buffered.



Note Buffering URLs longer than 1159 bytes is only supported for the Websense filtering server.

Step 2 To configure the maximum memory available for buffering pending URLs (and for buffering long URLs with Websense), enter the following command:

```
hostname(config)# url-block url-mempool memory-pool-size
```

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 FWSM 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 FWSM 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:

```
hostname(config)# 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, page 16-7](#)
- [Enabling Filtering of Long HTTP URLs, page 16-7](#)
- [Truncating Long HTTP URLs, page 16-7](#)

- [Exempting Traffic from Filtering, page 16-8](#)

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 FWSM allows the reply from the web server to reach the originating client. If the filtering server denies the request, the FWSM redirects the user to a block page, indicating that access was denied.

To enable HTTP filtering, enter the following command:

```
hostname(config)# filter url {http | port[-port] | except} local_ip local_mask foreign_ip
foreign_mask [allow][cgi-truncate][longurl-deny][longurl-truncate][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.

To create an exception to a previous filter condition, specify the keyword **except**.



Note

The filter exception rule works only when you use the default port.

The **allow** option causes the FWSM 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 FWSM considers an HTTP URL to be a long URL if it is greater than 1159 characters. For Websense servers, you can increase the maximum length allowed.

(Websense only) Configure the maximum size of a single URL with the following command:

```
hostname(config)# url-block url-size long_url_size
```

Replace *long_url_size* with a value from 2 to 4 for a maximum URL size of 2 KB to 4 KB. The default value is 2.

(Websense only) You can also configure the maximum size of the URL buffer memory pool with the following command:

```
hostname(config)# url-block url-mempool memory_pool_size
```

Replace *memory_pool_size* with a value from 2 to 10240 for a URL buffer memory pool size of 2 KB to 10,240 KB.

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 FWSM to truncate a long URL by entering the following command:

```
hostname(config)# filter url [longurl-truncate | longurl-deny | cgi-truncate]
```

The **longurl-truncate** option causes the FWSM 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 firewall performance.

Exempting Traffic from Filtering

To exempt specific traffic from filtering, enter the following command:

```
hostname(config)# filter url except source_ip source_mask dest_ip dest_mask
```

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

Secure Computing SmartFilter (formerly N2H2) does not support HTTPS filtering.

Because HTTPS content is encrypted, the FWSM sends the URL lookup without directory and filename information. When the filtering server approves an HTTPS connection request, the FWSM 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 FWSM 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 FWSM does not provide an authentication prompt for HTTPS, so a user must authenticate with the FWSM using HTTP or FTP before accessing HTTPS servers.

To enable HTTPS filtering, enter the following command:

```
hostname(config)# filter https port localIP local_mask foreign_IP foreign_mask [allow]
```

Replace *port* with the port number if a different port than the default port for HTTPS (443) is used. The filter exception rule works only when you use the default port.



Note

Because both HTTPS and HTTP traffic have the same GET request, the HTTPS protocol inspector will also filter HTTP traffic on the port number that you specify.

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 FWSM 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

Secure Computing SmartFilter (formerly known as N2H2) does not support FTP filtering.

When the filtering server approves an FTP connection request, the FWSM 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 FWSM changes code 250 to “550 Requested file is prohibited by URL filtering policy.”

To enable FTP filtering, enter the following command:

```
hostname(config)# filter ftp {port[-port] | except} localIP local_mask foreign_IP
foreign_mask [allow] [interact-block]
```

Replace *port* with the port number 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.

To create an exception to a previous filter condition, specify the keyword **except**.



Note

The filter exception rule works only when you use the default port.

The **allow** option causes the FWSM to forward FTP 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, page 16-10](#)
- [Viewing Buffer Configuration and Statistics, page 16-10](#)
- [Viewing Caching Statistics, page 16-11](#)
- [Viewing Filtering Performance Statistics, page 16-11](#)
- [Viewing Filtering Configuration, page 16-11](#)

Viewing Filtering Server Statistics

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

```
hostname# show running-config url-server
```

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

```
hostname# show running-config 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:

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

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

```
hostname# show url-server statistics
URL Server Statistics:
-----
Vendor                               websense
URLs total/allowed/denied            50/35/15
HTTPSs total/allowed/denied          1/1/0
FTPs total/allowed/denied            3/1/2

URL Server Status:
-----
10.130.28.18                          UP

URL Packets Sent and Received Stats:
-----
Message           Sent      Received
STATUS_REQUEST    65155    34773
LOOKUP_REQUEST    0         0
LOG_REQUEST       0         NA
-----
```

Viewing Buffer Configuration and Statistics

The **show running-config 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 running-config url-block** command:

```
hostname# show running-config 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** command:

```

hostname# show url-cache
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 running-config filter** command:

```

hostname# show running-config filter
filter url http 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

```

