Inspection of Basic Internet Protocols

The following topics explain application inspection for basic Internet protocols. For information on why you need to use inspection for certain protocols, and the overall methods for applying inspection, see Getting Started with Application Layer Protocol Inspection, page 6-1.

- DNS Inspection, page 7-1
- FTP Inspection, page 7-8
- HTTP Inspection, page 7-14
- ICMP Inspection, page 7-21
- ICMP Error Inspection, page 7-21
- Instant Messaging Inspection, page 7-21
- IP Options Inspection, page 7-26
- IPsec Pass Through Inspection, page 7-30
- IPv6 Inspection, page 7-33
- NetBIOS Inspection, page 7-37
- PPTP Inspection, page 7-39
- SMTP and Extended SMTP Inspection, page 7-39
- TFTP Inspection, page 7-45

DNS Inspection

The following sections describe DNS application inspection.

- DNS Inspection Actions, page 7-2
- Defaults for DNS Inspection, page 7-2
- Configure DNS Inspection, page 7-2
- Monitoring DNS Inspection, page 7-8
DNS Inspection

DNS Inspection Actions

DNS inspection is enabled by default. You can customize DNS inspection to perform many tasks:

- Translate the DNS record based on the NAT configuration. For more information, see DNS and NAT, page 5-21.
- Enforce message length, domain-name length, and label length.
- Verify the integrity of the domain-name referred to by the pointer if compression pointers are encountered in the DNS message.
- Check to see if a compression pointer loop exists.
- Inspect packets based on the DNS header, type, class and more.

Defaults for DNS Inspection

DNS inspection is enabled by default, using the preset_dns_map inspection class map:

- The maximum DNS message length is 512 bytes.
- The maximum client DNS message length is automatically set to match the Resource Record.
- DNS Guard is enabled, so the ASA tears down the DNS session associated with a DNS query as soon as the DNS reply is forwarded by the ASA. The ASA also monitors the message exchange to ensure that the ID of the DNS reply matches the ID of the DNS query.
- Translation of the DNS record based on the NAT configuration is enabled.
- Protocol enforcement is enabled, which enables DNS message format check, including domain name length of no more than 255 characters, label length of 63 characters, compression, and looped pointer check.

See the following default DNS inspection commands:

```plaintext
class-map inspection_default
match default-inspection-traffic
policy-map type inspect dns preset_dns_map
parameters
    message-length maximum client auto
    message-length maximum 512
dns-guard
protocol-enforcement
    nat-rewrite
policy-map global_policy
class inspection_default
    inspect dns preset_dns_map
    ! ...
service-policy global_policy global
```

Configure DNS Inspection

DNS inspection is enabled by default. You need to configure it only if you want non-default processing. If you want to customize DNS inspection, use the following process.
Configure DNS Inspection Policy Map

You can create a DNS inspection policy map to customize DNS inspection actions if the default inspection behavior is not sufficient for your network.

When defining traffic matching criteria, you can either create a class map or include the match statements directly in the policy map. The following procedure explains both approaches.

Before You Begin
Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create a DNS inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify match commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the match not command. For example, if the match not command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each match command, you should identify the traffic directly in the policy map.

a. Create the class map by entering the following command:

```
hostname(config)# class-map type inspect dns [match-all | match-any] class_map_name
hostname(config-cmap)#
```

Where the class_map_name is the name of the class map. The match-all keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one match statement. The CLI enters class-map configuration mode, where you can enter one or more match commands.

b. (Optional) To add a description to the class map, enter the following command:

```
hostname(config-cmap)# description string
```

Where string is the description of the class map (up to 200 characters).

c. Specify the traffic on which you want to perform actions using one of the following match commands. If you use a match not command, then any traffic that does not match the criterion in the match not command has the action applied.
**Chapter 7      Inspection of Basic Internet Protocols**

**DNS Inspection**

- **match [not] header-flag [eq] [f_name [f_name...]] [f_value]**—Matches the DNS flag. The `f_name` argument is the DNS flag name, one of the following: AA (Authoritative Answer), QR (Query), RA (Recursion Available), RD (Recursion Desired), TC (Truncation). The `f_value` argument is the 16-bit value in hex starting with 0x, from 0x0 to 0xffff. The `eq` keyword specifies an exact match (match all); without the `eq` keyword, the packet only needs to match one of the specified headers (match any). For example, **match header-flag AA QR**.

- **match [not] dns-type [eq] [t_name | t_value1 t_value2]**—Matches the DNS type. The `t_name` argument is the DNS type name, one of the following: A (IPv4 address), AXFR (full zone transfer), CNAME (canonical name), IXFR (incremental zone transfer), NS (authoritative name server), SOA (start of a zone of authority) or TSIG (transaction signature). The `t_value` arguments are arbitrary values in the DNS type field (0-65535). The `range` keyword specifies a range, and the `eq` keyword specifies an exact match. For example: **match dns-type eq A**.

- **match [not] dns-class [eq] [in | c_value1 c_value2]**—Matches the DNS class. The class is either `in` (for Internet) or `c_value`, an arbitrary value from 0 to 65535 in the DNS class field. The `range` keyword specifies a range, and the `eq` keyword specifies an exact match. For example: **match dns-class eq in**.

- **match [not] [question | resource-record [answer | authority | additional]]**—Matches a DNS question or resource record. The `question` keyword specifies the question portion of a DNS message. The `resource-record` keyword specifies one of these sections of the resource record: answer, authority, or additional. For example: **match resource-record answer**.

- **match [not] domain-name regex [regex_name | class class_name]**—Matches the DNS message domain name list against the specified regular expression or regular expression class.

d. Enter `exit` to leave class map configuration mode.

**Step 2**  
Create a DNS inspection policy map, enter the following command:

```
hostname(config)# policy-map type inspect dns policy_map_name
```

Where the `policy_map_name` is the name of the policy map. The CLI enters policy-map configuration mode.

**Step 3**  
(Optional) To add a description to the policy map, enter the following command:

```
hostname(config-pmap)# description string
```

**Step 4**  
To apply actions to matching traffic, perform the following steps.

a. Specify the traffic on which you want to perform actions using one of the following methods:

- If you created a DNS class map, specify it by entering the following command:

  ```
  hostname(config-pmap-c)# class class_map_name
  ```

- Specify traffic directly in the policy map using one of the `match` commands described for DNS class maps. If you use a `match not` command, then any traffic that does not match the criterion in the `match not` command has the action applied.

b. Specify the action you want to perform on the matching traffic by entering the following command:

```
hostname(config-pmap-c)# {drop [log] | drop-connection [log] | enforce-tsig [{drop} [log]} | mask [log] | log}
```

Not all options are available for each `match` or `class` command. See the CLI help or the command reference for the exact options available.

The `drop` keyword drops all packets that match.
The **drop-connection** keyword drops the packet and closes the connection.

The **mask** keyword masks out the matching portion of the packet. This action is available for header flag matches only.

The **log** keyword, which you can use alone or with one of the other keywords, sends a system log message.

The **enforce-tsig** ([drop] [log]) keyword enforces the presence of the TSIG resource record in a message. You can drop a packet without the TSIG resource record, log it, or drop and log it. You can use this option in conjunction with the mask action for header flag matches; otherwise, this action is exclusive with the other actions.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see Defining Actions in an Inspection Policy Map, page 2-4.

For example:

```
hostname(config)# policy-map type inspect dns dns-map
hostname(config-pmap)# class dns-class-map
hostname(config-pmap-c)# drop
hostname(config-pmap-c)# match header-flag eq aa
hostname(config-pmap-c)# drop log
```

**Step 5**
To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b. Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **dns-guard**—Enables DNS Guard. The ASA tears down the DNS session associated with a DNS query as soon as the DNS reply is forwarded by the ASA. The ASA also monitors the message exchange to ensure that the ID of the DNS reply matches the ID of the DNS query.

- **id-mismatch count number duration seconds action log**—Enables logging for excessive DNS ID mismatches, where the **count number duration seconds action log** arguments specify the maximum number of mismatch instances per second before a system message log is sent.

- **id-randomization**—Randomizes the DNS identifier for a DNS query.

- **message-length maximum {length | client {length | auto} | server {length | auto}}**—Sets the maximum DNS message length, from 512 to 65535 bytes. You can also set the maximum length for client or server messages. The **auto** keyword sets the maximum length to the value in the Resource Record.

- **nat-rewrite**—Translates the DNS record based on the NAT configuration.

- **protocol-enforcement**—Enables DNS message format check, including domain name length of no more than 255 characters, label length of 63 characters, compression, and looped pointer check.

- **tsig enforced action {[drop] [log]}**—Requires a TSIG resource record to be present. You can drop a non-conforming packet, log the packet, or both.

For example:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)# dns-guard
hostname(config-pmap-p)# message-length maximum 1024
hostname(config-pmap-p)# nat-rewrite
hostname(config-pmap-p)# protocol-enforcement
```
Example
The following example shows how to define a DNS inspection policy map.

```plaintext
regex domain_example "example\.com"
regex domain_foo "foo\.com"

! define the domain names that the server serves
class-map type inspect regex match-any my_domains
  match regex domain_example
  match regex domain_foo

! Define a DNS map for query only
class-map type inspect dns match-all pub_server_map
  match not header-flag QR
  match question
  match not domain-name regex class my_domains

policy-map type inspect dns new_dns_map
class pub_server_map
  drop log
  match header-flag RD
  mask log
  parameters
    message-length maximum client auto
    message-length maximum 512
    dns-guard
    protocol-enforcement
    nat-rewrite
```

Configure the DNS Inspection Service Policy

The default ASA configuration includes DNS inspection on the default port applied globally on all interfaces. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

**Step 1**  If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```plaintext
class-map name
match parameter
```

Example:

```plaintext
hostname(config)# class-map dns_class_map
hostname(config-cmap)# match access-list dns
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see [Identify Traffic (Layer 3/4 Class Maps), page 1-13](#).

**Step 2**  Add or edit a policy map that sets the actions to take with the class map traffic.
**DNS Inspection**

**Step 3** Identify the L3/L4 class map you are using for DNS inspection.

```bash
class name
```

Example:

```bash
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify `inspection_default` for the `name`. Otherwise, you are specifying the class you created earlier in this procedure.

**Step 4** Configure DNS inspection.

```bash
inspect dns [dns_policy_map] [dynamic-filter-snoop]
```

Where:

- `dns_policy_map` is the optional DNS inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the DNS inspection policy map, see `Configure DNS Inspection Policy Map`, page 7-3.

- `dynamic-filter-snoop` enables dynamic filter snooping, which is used exclusively by the Botnet Traffic Filter. Include this keyword only if you use Botnet Traffic Filtering. We suggest that you enable DNS snooping only on interfaces where external DNS requests are going. Enabling DNS snooping on all UDP DNS traffic, including that going to an internal DNS server, creates unnecessary load on the ASA.

Example:

```bash
hostname(config-class)# no inspect dns
hostname(config-class)# inspect dns dns-map
```

**Note** If you are editing the default global policy (or any in-use policy) to use a different DNS inspection policy map (for example, you are replacing the default preset_dns_map), you must remove the DNS inspection with the `no inspect dns` command, and then re-add it with the new DNS inspection policy map name.

**Step 5** If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

```bash
service-policy policymap_name {global | interface interface_name}
```

Example:

```bash
hostname(config)# service-policy global_policy global
```

The `global` keyword applies the policy map to all interfaces, and `interface` applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.
Examples
The following example shows how to use a new inspection policy map in the global default configuration:

```plaintext
policy-map global_policy
class inspection_default
  no inspect dns preset_dns_map
  inspect dns new_dns_map
service-policy global_policy global
```

Monitoring DNS Inspection

To view information about the current DNS connections, enter the following command:

```
hostname# show conn
```

For connections using a DNS server, the source port of the connection may be replaced by the IP address of the DNS server in the show conn command output.

A single connection is created for multiple DNS sessions, as long as they are between the same two hosts, and the sessions have the same 5-tuple (source/destination IP address, source/destination port, and protocol). DNS identification is tracked by app_id, and the idle timer for each app_id runs independently.

Because the app_id expires independently, a legitimate DNS response can only pass through the security appliance within a limited period of time and there is no resource build-up. However, when you enter the `show conn` command, you see the idle timer of a DNS connection being reset by a new DNS session. This is due to the nature of the shared DNS connection and is by design.

To display the statistics for DNS application inspection, enter the `show service-policy` command. The following is sample output from the `show service-policy` command:

```
hostname# show service-policy
Interface outside:
  Service-policy: sample_policy
    Class-map: dns_port
      Inspect: dns maximum-length 1500, packet 0, drop 0, reset-drop 0
```

FTP Inspection

The following sections describe the FTP inspection engine.

- FTP Inspection Overview, page 7-8
- Strict FTP, page 7-9
- Configure FTP Inspection, page 7-10
- Verifying and Monitoring FTP Inspection, page 7-14

FTP Inspection Overview

The FTP application inspection inspects the FTP sessions and performs four tasks:

- Prepares dynamic secondary data connection
Chapter 7  Inspection of Basic Internet Protocols

FTP Inspection

- Tracks the FTP command-response sequence
- Generates an audit trail
- Translates the embedded IP address

FTP application inspection prepares secondary channels for FTP data transfer. Ports for these channels are negotiated through PORT or PASV commands. The channels are allocated in response to a file upload, a file download, or a directory listing event.

Note

If you disable FTP inspection engines with the `no inspect ftp` command, outbound users can start connections only in passive mode, and all inbound FTP is disabled.

Strict FTP

Strict FTP increases the security of protected networks by preventing web browsers from sending embedded commands in FTP requests. To enable strict FTP, include the `strict` option with the `inspect ftp` command.

When you use strict FTP, you can optionally specify an FTP inspection policy map to specify FTP commands that are not permitted to pass through the ASA.

After you enable the `strict` option on an interface, FTP inspection enforces the following behavior:

- An FTP command must be acknowledged before the ASA allows a new command.
- The ASA drops connections that send embedded commands.
- The 227 and PORT commands are checked to ensure they do not appear in an error string.

Caution

Using the `strict` option may cause the failure of FTP clients that are not strictly compliant with FTP RFCs.

If the `strict` option is enabled, each FTP command and response sequence is tracked for the following anomalous activity:

- Truncated command—Number of commas in the PORT and PASV reply command is checked to see if it is five. If it is not five, then the PORT command is assumed to be truncated and the TCP connection is closed.
- Incorrect command—Checks the FTP command to see if it ends with `<CR><LF>` characters, as required by the RFC. If it does not, the connection is closed.
- Size of RETR and STOR commands—These are checked against a fixed constant. If the size is greater, then an error message is logged and the connection is closed.
- Command spoofing—The PORT command should always be sent from the client. The TCP connection is denied if a PORT command is sent from the server.
- Reply spoofing—PASV reply command (227) should always be sent from the server. The TCP connection is denied if a PASV reply command is sent from the client. This prevents the security hole when the user executes "227 xxxxx a1, a2, a3, a4, p1, p2."
- TCP stream editing—The ASA closes the connection if it detects TCP stream editing.
- Invalid port negotiation—The negotiated dynamic port value is checked to see if it is less than 1024. As port numbers in the range from 1 to 1024 are reserved for well-known connections, if the negotiated port falls in this range, then the TCP connection is freed.
FTP Inspection

Command pipelining—The number of characters present after the port numbers in the PORT and PASV reply command is cross checked with a constant value of 8. If it is more than 8, then the TCP connection is closed.

The ASA replaces the FTP server response to the SYST command with a series of Xs. to prevent the server from revealing its system type to FTP clients. To override this default behavior, use the `no mask-syst-reply` command in the FTP map.

Configure FTP Inspection

FTP inspection is enabled by default. You need to configure it only if you want non-default processing. If you want to customize FTP inspection, use the following process.

**Procedure**

**Step 1** Configure an FTP Inspection Policy Map, page 7-10.

**Step 2** Configure the FTP Inspection Service Policy, page 7-13.

Configure an FTP Inspection Policy Map

FTP command filtering and security checks are provided using strict FTP inspection for improved security and control. Protocol conformance includes packet length checks, delimiters and packet format checks, command terminator checks, and command validation.

Blocking FTP based on user values is also supported so that it is possible for FTP sites to post files for download, but restrict access to certain users. You can block FTP connections based on file type, server name, and other attributes. System message logs are generated if an FTP connection is denied after inspection.

If you want FTP inspection to allow FTP servers to reveal their system type to FTP clients, and limit the allowed FTP commands, then create and configure an FTP inspection policy map. You can then apply the map when you enable FTP inspection.

**Before You Begin**

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

**Procedure**

**Step 1** (Optional) Create an FTP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify `match` commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the `match not` command. For example, if the `match not` command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.
For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each match command, you should identify the traffic directly in the policy map.

a. Create the class map by entering the following command:

```
hostname(config)# class-map type inspect ftp [match-all | match-any] class_map_name
hostname(config-cmap)#
```

Where the class_map_name is the name of the class map. The match-all keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one match statement. The CLI enters class-map configuration mode, where you can enter one or more match commands.

b. (Optional) To add a description to the class map, enter the following command:

```
hostname(config-cmap)# description string
```

Where string is the description of the class map (up to 200 characters).

c. Specify the traffic on which you want to perform actions using one of the following match commands. If you use a match not command, then any traffic that does not match the criterion in the match not command has the action applied.

- **match [not] filename regex {regex_name | class class_name}**—Matches the filename in the FTP transfer against the specified regular expression or regular expression class.
- **match [not] filetype regex {regex_name | class class_name}**—Matches the file type in the FTP transfer against the specified regular expression or regular expression class.
- **match [not] request-command ftp_command [ftp_command...]**—Matches the FTP command, one or more of the following:
  - APPE—Append to a file.
  - CDUP—Changes to the parent directory of the current working directory.
  - DELE—Deletes a file on the server.
  - GET—Gets a file from the server.
  - HELP—Provides help information.
  - MKD—Makes a directory on the server.
  - PUT—Sends a file to the server.
  - RMD—Deletes a directory on the server.
  - RNFR—Specifies the “rename-from” filename.
  - RNTO—Specifies the “rename-to” filename.
  - SITE—Used to specify a server-specific command. This is usually used for remote administration.
  - STOU—Stores a file using a unique file name.
- **match [not] server regex {regex_name | class class_name}**—Matches the FTP server name against the specified regular expression or regular expression class.
- **match [not] username regex {regex_name | class class_name}**—Matches the FTP username against the specified regular expression or regular expression class.

d. Enter exit to leave class map configuration mode.
Step 2  Create an FTP inspection policy map:

```plaintext
hostname(config)# policy-map type inspect ftp policy_map_name
hostname(config-pmap)#
```

Where the `policy_map_name` is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3  (Optional) To add a description to the policy map, enter the following command:

```plaintext
hostname(config-pmap)# description string
```

Step 4  To apply actions to matching traffic, perform the following steps.

a. Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an FTP class map, specify it by entering the following command:
  ```plaintext
  hostname(config-pmap-c)# class class_map_name
  hostname(config-pmap-c)#
  ```

- Specify traffic directly in the policy map using one of the `match` commands described for FTP class maps. If you use a `match not` command, then any traffic that does not match the criterion in the `match not` command has the action applied.

b. Specify the action you want to perform on the matching traffic by entering the following command:

```plaintext
hostname(config-pmap-c)# reset [log]
```

The `reset` keyword drops the packet, closes the connection, and sends a TCP reset to the server or client. Add the `log` keyword to send a system log message.

You can specify multiple `class` or `match` commands in the policy map. For information about the order of `class` and `match` commands, see Defining Actions in an Inspection Policy Map, page 2-4.

Step 5  To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```plaintext
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b. Set one or more parameters. You can set the following options; use the `no` form of the command to disable the option:

- `mask-banner`—Masks the greeting banner from the FTP server.
- `mask-syst-reply`—Masks the reply to `syst` command.

Example

Before submitting a username and password, all FTP users are presented with a greeting banner. By default, this banner includes version information useful to hackers trying to identify weaknesses in a system. The following example shows how to mask this banner:

```plaintext
hostname(config)# policy-map type inspect ftp mymap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# mask-banner

hostname(config)# class-map match-all ftp-traffic
hostname(config-cmap)# match port tcp eq ftp

hostname(config)# policy-map ftp-policy
hostname(config-pmap)# class ftp-traffic
```
Configure the FTP Inspection Service Policy

The default ASA configuration includes FTP inspection on the default port applied globally on all interfaces. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

Step 1 If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map ftp_class_map
hostname(config-cmap)# match access-list ftp
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (match default-inspection-traffic). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic.

```
policy-map name
```

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3 Identify the L3/L4 class map you are using for FTP inspection.

```
class name
```

Example:

```
hostname(config-pmap-c)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify inspection_default for the name. Otherwise, you are specifying the class you created earlier in this procedure.

Step 4 Configure FTP inspection.

```
inspect ftp [strict [ftp_policy_map]]
```

Where:

- strict implements strict FTP. You must use strict FTP to specify an FTP inspection policy map.
• \textit{ftp\_policy\_map} is the optional FTP inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the FTP inspection policy map, see \textit{Configure an FTP Inspection Policy Map}, page 7-10.

Example:

\begin{verbatim}
hostname(config-class)# no inspect ftp
hostname(config-class)# inspect ftp strict ftp-map
\end{verbatim}

\textbf{Note} If you are editing the default global policy (or any in-use policy) to use a different FTP inspection policy map, you must remove the FTP inspection with the \texttt{no inspect ftp} command, and then re-add it with the new FTP inspection policy map name.

\textbf{Step 5} If you are editing an existing service policy (such as the default global policy called \texttt{global\_policy}), you are done. Otherwise, activate the policy map on one or more interfaces.

\begin{verbatim}
service-policy policymap_name {global | interface interface_name}
\end{verbatim}

Example:

\begin{verbatim}
hostname(config)# service-policy global_policy global
\end{verbatim}

The \texttt{global} keyword applies the policy map to all interfaces, and \texttt{interface} applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

\section*{Verifying and Monitoring FTP Inspection}

FTP application inspection generates the following log messages:

• An Audit record 303002 is generated for each file that is retrieved or uploaded.

• The FTP command is checked to see if it is RETR or STOR and the retrieve and store commands are logged.

• The username is obtained by looking up a table providing the IP address.

• The username, source IP address, destination IP address, NAT address, and the file operation are logged.

• Audit record 201005 is generated if the secondary dynamic channel preparation failed due to memory shortage.

In conjunction with NAT, the FTP application inspection translates the IP address within the application payload. This is described in detail in RFC 959.

\section*{HTTP Inspection}

The following sections describe the HTTP inspection engine.

• \textit{HTTP Inspection Overview}, page 7-15

• \textit{Configure HTTP Inspection}, page 7-15
HTTP Inspection Overview

Tip
You can install a service module that performs application and URL filtering, which includes HTTP inspection, such as ASA CX or ASA FirePOWER. The HTTP inspection running on the ASA is not compatible with these modules. Note that it is far easier to configure application filtering using a purpose-built module rather than trying to manually configure it on the ASA using an HTTP inspection policy map.

Use the HTTP inspection engine to protect against specific attacks and other threats that are associated with HTTP traffic.

HTTP application inspection scans HTTP headers and body, and performs various checks on the data. These checks prevent various HTTP constructs, content types, and tunneling and messaging protocols from traversing the security appliance.

The enhanced HTTP inspection feature, which is also known as an application firewall and is available when you configure an HTTP inspection policy map, can help prevent attackers from using HTTP messages for circumventing network security policy.

HTTP application inspection can block tunneled applications and non-ASCII characters in HTTP requests and responses, preventing malicious content from reaching the web server. Size limiting of various elements in HTTP request and response headers, URL blocking, and HTTP server header type spoofing are also supported.

Enhanced HTTP inspection verifies the following for all HTTP messages:

- Conformance to RFC 2616
- Use of RFC-defined methods only.
- Compliance with the additional criteria.

Configure HTTP Inspection

HTTP inspection is not enabled by default. If you are not using a purpose-built module for HTTP inspection and application filtering, such as ASA CX or ASA FirePOWER, you can manually configure HTTP inspection on the ASA using the following process.

Tip
Do not configure HTTP inspection in both a service module and on the ASA, as the inspections are not compatible.

Procedure

Step 1 Configure an HTTP Inspection Policy Map, page 7-16.

Step 2 Configure the HTTP Inspection Service Policy, page 7-19.
Configure an HTTP Inspection Policy Map

To specify actions when a message violates a parameter, create an HTTP inspection policy map. You can then apply the inspection policy map when you enable HTTP inspection.

Before You Begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1

(Optional) Create an HTTP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify match commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the match not command. For example, if the match not command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each match command, you should identify the traffic directly in the policy map.

a. Create the class map by entering the following command:

```
hostname(config)# class-map type inspect http [match-all | match-any] class_map_name
```

Where the class_map_name is the name of the class map. The match-all keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one match statement. The CLI enters class-map configuration mode, where you can enter one or more match commands.

b. (Optional) To add a description to the class map, enter the following command:

```
hostname(config-cmap)# description string
```

Where string is the description of the class map (up to 200 characters).

c. Specify the traffic on which you want to perform actions using one of the following match commands. If you use a match not command, then any traffic that does not match the criterion in the match not command has the action applied.

- **match [not] req-resp content-type mismatch**—Matches traffic with a content-type field in the HTTP response that does not match the accept field in the corresponding HTTP request message.
- **match [not] request args regex \{regex_name | class class_name\}**—Matches text found in the HTTP request message arguments against the specified regular expression or regular expression class.
- **match [not] request body \{regex \{regex_name | class class_name\} \| length gt bytes\}**—Matches text found in the HTTP request message body against the specified regular expression or regular expression class, or messages where the request body is greater than the specified length.
HTTP Inspection

- **match [not] request header {field | regex regex_name} regex {regex_name | class class_name}**—Matches the content of a field in the HTTP request message header against the specified regular expression or regular expression class. You can specify the field name explicitly or match the field name to a regular expression. Field names are: accept, accept-charset, accept-encoding, accept-language, allow, authorization, cache-control, connection, content-encoding, content-language, content-length, content-location, content-md5, content-range, content-type, cookie, date, expect, expires, from, host, if-match, if-modified-since, if-none-match, if-range, if-unmodified-since, last-modified, max-forwards, pragma, proxy-authentication, range, referer, te, trailer, transfer-encoding, upgrade, user-agent, via, warning.

- **match [not] request header {field | regex {regex_name | class class_name}} {length gt bytes | count gt number}**—Matches the length of the specified fields in the HTTP request message header, or the overall number of fields (count) in the header. You can specify the field name explicitly or match the field name to a regular expression or regular expression class. Field names are listed in the previous bullet.

- **match [not] request header {length gt bytes | count gt number | non-ascii}**—Matches the overall length of the HTTP request message header, or the overall number of fields (count) in the header, or headers that have non-ASCII characters.

- **match [not] request method {method | regex {regex_name | class class_name}}**—Matches the HTTP request method. You can specify the method explicitly or match the method to a regular expression or regular expression class. Methods are: bcopy, bdelete, bmove, bpropfind, bpropatch, connect, copy, delete, edit, get, getattribute, getattributenames, getproperties, head, index, lock, mkcol, mkdir, move, notify, options, poll, post, propfind, proppatch, put, revadd, revlabel, revlog, revnum, save, search, setattribute, startrev, stoptrev, subscribe, trace, unedit, unlock, unsubscribe.

- **match [not] request uri {regex {regex_name | class class_name}} length gt bytes**—Matches text found in the HTTP request message URI against the specified regular expression or regular expression class, or messages where the request URI is greater than the specified length.

- **match [not] response body {active-x | java-applet | regex {regex_name | class class_name}}**—Matches text found in the HTTP response message body against the specified regular expression or regular expression class, or comments out Java applet and Active X object tags in order to filter them.

- **match [not] response body length gt bytes**—Matches HTTP response messages where the body is greater than the specified length.

- **match [not] response header {field | regex regex_name} regex {regex_name | class class_name}**—Matches the content of a field in the HTTP response message header against the specified regular expression or regular expression class. You can specify the field name explicitly or match the field name to a regular expression. Field names are: accept-ranges, age, allow, cache-control, connection, content-encoding, content-language, content-length, content-location, content-md5, content-range, content-type, date, etag, expires, last-modified, location, pragma, proxy-authenticate, retry-after, server, set-cookie, trailer, transfer-encoding, upgrade, vary, via, warning, www-authenticate.

- **match [not] response header {field | regex {regex_name | class class_name}} {length gt bytes | count gt number}**—Matches the length of the specified fields in the HTTP response message header, or the overall number of fields (count) in the header. You can specify the field name explicitly or match the field name to a regular expression or regular expression class. Field names are listed in the previous bullet.
HTTP Inspection

- **match [not] response header {length gt bytes | count gt number | non-ascii}**—Matches the overall length of the HTTP response message header, or the overall number of fields (count) in the header, or headers that have non-ASCII characters.

- **match [not] response status-line regex {regex_name | class class_name}**—Matches text found in the HTTP response message status line against the specified regular expression or regular expression class.

d. Enter **exit** to leave class map configuration mode.

**Step 2** Create an HTTP inspection policy map:

```
hostname(config)# policy-map type inspect http policy_map_name
hostname(config-pmap)#
```

Where the **policy_map_name** is the name of the policy map. The CLI enters policy-map configuration mode.

**Step 3** (Optional) To add a description to the policy map, enter the following command:

```
hostname(config-pmap)# description string
```

**Step 4** To apply actions to matching traffic, perform the following steps.

a. Specify the traffic on which you want to perform actions using one of the following methods:

   - If you created an HTTP class map, specify it by entering the following command:
     
     ```
     hostname(config-pmap)# class class_map_name
     hostname(config-pmap-c)#
     ```

   - Specify traffic directly in the policy map using one of the **match** commands described for HTTP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b. Specify the action you want to perform on the matching traffic by entering the following command:

   ```
   hostname(config-pmap-c)# {drop-connection [log] | reset [log] | log}
   ```

   The **drop-connection** keyword drops the packet and closes the connection.

   The **reset** keyword drops the packet, closes the connection, and sends a TCP reset to the server or client.

   The **log** keyword, which you can use alone or with one of the other keywords, sends a system log message.

   You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see Defining Actions in an Inspection Policy Map, page 2-4.

**Step 5** To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

   ```
   hostname(config-pmap)# parameters
   hostname(config-pmap-p)#
   ```

b. Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

   - **body-match-maximum number**—Sets the maximum number of characters in the body of an HTTP message that should be searched in a body match. The default is 200 bytes. A large number will have a significant impact on performance.
- `protocol-violation action [drop-connection [log] | reset [log] | log]`—Sets the maximum number of characters in the body of an HTTP message that should be searched in a body match. The default is 200 bytes. A large number will have a significant impact on performance.

- Checks for HTTP protocol violations. You must also choose the action to take for violations (drop connection, reset, or log) and whether to enable or disable logging.

- `spoof-server string`—Substitutes a string for the server header field. WebVPN streams are not subject to the `spoof-server` command.

---

**Example**

The following example shows how to define an HTTP inspection policy map that will allow and log any HTTP connection that attempts to access “www.xyz.com/.*.asp” or “www.xyz[0-9][0-9].com” with methods “GET” or “PUT.” All other URL/Method combinations will be silently allowed.

```bash
hostname(config)# regex url1 "www.xyz.com/.*.asp"
hostname(config)# regex url2 "www.xyz[0-9][0-9].com"
hostname(config)# regex get "GET"
hostname(config)# regex put "PUT"

hostname(config)# class-map type regex match-any url_to_log
hostname(config-cmap)# match regex url1
hostname(config-cmap)# match regex url2
hostname(config-cmap)# exit

hostname(config)# class-map type regex match-any methods_to_log
hostname(config-cmap)# match regex get
hostname(config-cmap)# match regex put
hostname(config-cmap)# exit

hostname(config)# class-map type inspect http http_url_policy
hostname(config-cmap)# match request uri regex class url_to_log
hostname(config-cmap)# match request method regex class methods_to_log
hostname(config-cmap)# exit

hostname(config)# policy-map type inspect http http_policy
hostname(config-pmap)# class http_url_policy
hostname(config-pmap-c)# log
```

---

**Configure the HTTP Inspection Service Policy**

HTTP inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default HTTP ports, so you can simply edit the default global inspection policy to add HTTP inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

**Procedure**

**Step 1** If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```bash
class-map name
match parameter
```

**Example:**

```bash
hostname(config)# class-map http_class_map
hostname(config-cmap)# match access-list http
```
HTTP Inspection

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (match default-inspection-traffic). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

Step 2  
Add or edit a policy map that sets the actions to take with the class map traffic.

policy-map name

Example:
hostname(config)# policy-map global_policy

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3  
Identify the L3/L4 class map you are using for HTTP inspection.

class name

Example:
hostname(config-pmap)# class inspection_default

To edit the default policy, or to use the special inspection_default class map in a new policy, specify inspection_default for the name. Otherwise, you are specifying the class you created earlier in this procedure.

Step 4  
Configure HTTP inspection.

inspect http [http_policy_map]

Where http_policy_map is the optional HTTP inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the HTTP inspection policy map, see Configure an HTTP Inspection Policy Map, page 7-16.

Example:
hostname(config-class)# no inspect http
hostname(config-class)# inspect http http-map

Note  
If you are editing the default global policy (or any in-use policy) to use a different HTTP inspection policy map, you must remove the HTTP inspection with the no inspect http command, and then re-add it with the new HTTP inspection policy map name.

Step 5  
If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy policymap_name {global | interface interface_name}

Example:
hostname(config)# service-policy global_policy global

The global keyword applies the policy map to all interfaces, and interface applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.
ICMP Inspection

The ICMP inspection engine allows ICMP traffic to have a “session” so it can be inspected like TCP and UDP traffic. Without the ICMP inspection engine, we recommend that you do not allow ICMP through the ASA in an ACL. Without stateful inspection, ICMP can be used to attack your network. The ICMP inspection engine ensures that there is only one response for each request, and that the sequence number is correct.

However, ICMP traffic directed to an ASA interface is never inspected, even if you enable ICMP inspection. Thus, a ping (echo request) to an interface can fail under specific circumstances, such as when the echo request comes from a source that the ASA can reach through a backup default route.

For information on enabling ICMP inspection, see Configure Application Layer Protocol Inspection, page 6-9.

ICMP Error Inspection

When ICMP Error inspection is enabled, the ASA creates translation sessions for intermediate hops that send ICMP error messages, based on the NAT configuration. The ASA overwrites the packet with the translated IP addresses.

When disabled, the ASA does not create translation sessions for intermediate nodes that generate ICMP error messages. ICMP error messages generated by the intermediate nodes between the inside host and the ASA reach the outside host without consuming any additional NAT resource. This is undesirable when an outside host uses the traceroute command to trace the hops to the destination on the inside of the ASA. When the ASA does not translate the intermediate hops, all the intermediate hops appear with the mapped destination IP address.

The ICMP payload is scanned to retrieve the five-tuple from the original packet. Using the retrieved five-tuple, a lookup is performed to determine the original address of the client. The ICMP error inspection engine makes the following changes to the ICMP packet:

- In the IP Header, the mapped IP is changed to the real IP (Destination Address) and the IP checksum is modified.
- In the ICMP Header, the ICMP checksum is modified due to the changes in the ICMP packet.
- In the Payload, the following changes are made:
  - Original packet mapped IP is changed to the real IP
  - Original packet mapped port is changed to the real Port
  - Original packet IP checksum is recalculated

For information on enabling ICMP Error inspection, see Configure Application Layer Protocol Inspection, page 6-9.

Instant Messaging Inspection

The Instant Messaging (IM) inspect engine lets you control the network usage of IM and stop leakage of confidential data, propagation of worms, and other threats to the corporate network.

IM inspection is not enabled by default. You must configure it if you want IM inspection.
Configure an Instant Messaging Inspection Policy Map

To specify actions when a message violates a parameter, create an IM inspection policy map. You can then apply the inspection policy map when you enable IM inspection.

Before You Begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an IM inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify match commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the match not command. For example, if the match not command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each match command, you should identify the traffic directly in the policy map.

a. Create the class map by entering the following command:

```
hostname(config)# class-map type inspect im [match-all | match-any] class_map_name
```

Where the class_map_name is the name of the class map. The match-all keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one match statement. The CLI enters class-map configuration mode, where you can enter one or more match commands.

b. (Optional) To add a description to the class map, enter the following command:

```
hostname(config-cmap)# description string
```

Where string is the description of the class map (up to 200 characters).

c. Specify the traffic on which you want to perform actions using one of the following match commands. If you use a match not command, then any traffic that does not match the criterion in the match not command has the action applied.

- **match [not] protocol {im-yahoo | im-msn}**—Matches a specific IM protocol, either Yahoo or MSN.
Chapter 7  Inspection of Basic Internet Protocols

Instant Messaging Inspection

- **match [not] service {chat | file-transfer | webcam | voice-chat | conference | games}**—Matches the specific IM service.

- **match [not] login-name regex {regex_name | class class_name}**—Matches the source client login name of the IM message against the specified regular expression or regular expression class.

- **match [not] peer-login-name regex {regex_name | class class_name}**—Matches the destination peer login name of the IM message against the specified regular expression or regular expression class.

- **match [not] ip-address ip_address mask**—Matches the source IP address and mask of the IM message.

- **match [not] peer-ip-address ip_address mask**—Matches the destination IP address and mask of the IM message.

- **match [not] version regex {regex_name | class class_name}**—Matches the version of the IM message against the specified regular expression or regular expression class.

- **match [not] filename regex {regex_name | class class_name}**—Matches the filename of the IM message against the specified regular expression or regular expression class. This match is not supported for the MSN IM protocol.

d. Enter **exit** to leave class map configuration mode.

**Step 2** Create an IM inspection policy map:

```
hostname(config)# policy-map type inspect im policy_map_name
```

Where the **policy_map_name** is the name of the policy map. The CLI enters policy-map configuration mode.

**Step 3** (Optional) To add a description to the policy map, enter the following command:

```
hostname(config-pmap)# description string
```

**Step 4** To apply actions to matching traffic, perform the following steps.

a. Specify the traffic on which you want to perform actions using one of the following methods:

   i. If you created an IM class map, specify it by entering the following command:

      ```
      hostname(config-pmap)# class class_map_name
      hostname(config-pmap-c)#
      ```

   ii. Specify traffic directly in the policy map using one of the **match** commands described for IM class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b. Specify the action you want to perform on the matching traffic by entering the following command:

   ```
   hostname(config-pmap-c)# {drop-connection [log] | reset [log] | log}
   ```

   The **drop-connection** keyword drops the packet and closes the connection.

   The **reset** keyword drops the packet, closes the connection, and sends a TCP reset to the server or client.

   The **log** keyword, which you can use alone or with one of the other keywords, sends a system log message.
You can specify multiple `class` or `match` commands in the policy map. For information about the order of `class` and `match` commands, see Defining Actions in an Inspection Policy Map, page 2-4.

**Example**
The following example shows how to define an IM inspection policy map.

```
hostname(config)# regex loginname1 "ying\@yahoo.com"
hostname(config)# regex loginname2 "Kevin\@yahoo.com"
hostname(config)# regex loginname3 "rahul\@yahoo.com"
hostname(config)# regex loginname4 "darshant\@yahoo.com"
hostname(config)# regex yahoo_version_regex "1\0"
hostname(config)# regex gif_files ".\*\.gif"
hostname(config)# regex exe_files ".\*\.exe"

hostname(config)# class-map type regex match-any yahoo_src_login_name_regex
hostname(config-cmap)# match regex loginname1
hostname(config-cmap)# match regex loginname2

hostname(config)# class-map type regex match-any yahoo_dst_login_name_regex
hostname(config-cmap)# match regex loginname3
hostname(config-cmap)# match regex loginname4

hostname(config)# class-map type inspect im match-any yahoo_file_block_list
hostname(config-cmap)# match filename regex gif_files
hostname(config-cmap)# match filename regex exe_files

hostname(config)# class-map type inspect im match-all yahoo_im_policy
hostname(config-cmap)# match login-name regex class yahoo_src_login_name_regex
hostname(config-cmap)# match peer-login-name regex class yahoo_dst_login_name_regex

hostname(config)# class-map type inspect im match-all yahoo_im_policy2
hostname(config-cmap)# match version regex yahoo_version_regex

hostname(config)# class-map im_inspect_class_map
hostname(config-cmap)# match default-inspection-traffic

hostname(config)# policy-map type inspect im im_policy_all
hostname(config-pmap)# class yahoo_file_block_list
hostname(config-pmap-c)# match service file-transfer
hostname(config-pmap-c)# class yahoo_im_policy
hostname(config-pmap-c)# drop-connection
hostname(config-pmap)# class yahoo_im_policy2
hostname(config-pmap-c)# reset
hostname(config)# policy-map global_policy_name
hostname(config-pmap)# class im_inspect_class_map
hostname(config-pmap-c)# inspect im policy_all
```

**Configure the IM Inspection Service Policy**

IM inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default IM ports, so you can simply edit the default global inspection policy to add IM inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.
### Instant Messaging Inspection

#### Procedure

**Step 1**
If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```cisco
class-map name
match parameter
```

Example:

```cisco
hostname(config)# class-map im_class_map
hostname(config-cmap)# match access-list im
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (match default-inspection-traffic). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

**Step 2**
Add or edit a policy map that sets the actions to take with the class map traffic.

```cisco
policy-map name
```

Example:

```cisco
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

**Step 3**
Identify the L3/L4 class map you are using for IM inspection.

```cisco
class name
```

Example:

```cisco
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify **inspection_default** for the *name*. Otherwise, you are specifying the class you created earlier in this procedure.

**Step 4**
Configure IM inspection.

```cisco
inspect im [im_policy_map]
```

Where **im_policy_map** is the optional IM inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the IM inspection policy map, see Configure an Instant Messaging Inspection Policy Map, page 7-22.

Example:

```cisco
hostname(config-class)# no inspect im
hostname(config-class)# inspect im im-map
```

**Note**
If you are editing the default global policy (or any in-use policy) to use a different IM inspection policy map, you must remove the IM inspection with the **no inspect im** command, and then re-add it with the new IM inspection policy map name.

**Step 5**
If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

```cisco
service-policy policymap_name {global | interface interface_name}
```
Example:
hostname(config)# service-policy global_policy global

The *global* keyword applies the policy map to all interfaces, and *interface* applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

---

### IP Options Inspection

You can configure IP Options inspection to control which IP packets with specific IP options are allowed through the ASA. Configuring this inspection instructs the ASA to allow a packet to pass or to clear the specified IP options and then allow the packet to pass.

The following sections describe the IP Options inspection engine.

- IP Options Inspection Overview, page 7-26
- Defaults for IP Options Inspection, page 7-27
- Configure IP Options Inspection, page 7-27
- Monitoring IP Options Inspection, page 7-30

### IP Options Inspection Overview

Each IP packet contains an IP header with the Options field. The Options field, commonly referred to as IP Options, provide for control functions that are required in some situations but unnecessary for most common communications. In particular, IP Options include provisions for time stamps, security, and special routing. Use of IP Options is optional, and the field can contain zero, one, or more options.

For a list of IP options, with references to the relevant RFCs, see the IANA page, [http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml](http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml).

You can configure IP Options inspection to control which IP packets with specific IP options are allowed through the ASA. Configuring this inspection instructs the ASA to allow a packet to pass or to clear the specified IP options and then allow the packet to pass.

### What Happens When You Clear an Option

When you configure an IP options inspection policy map, you can specify whether you want to allow or clear each option type. If you do not specify an option type, packets that contain the option are dropped. If you simply allow an option, packets containing the option are passed through unchanged.

If you specify that you want to clear an option from IP headers, the IP header changes in the following ways:

- The option is removed from the header.
- The Options field is padded so that the field ends on a 32 bit boundary.
- Internet header length (IHL) in the packet changes.
- The total length of the packet changes.
- The checksum is recomputed.

**Supported IP Options for Inspection**

IP Options inspection can check for the following IP options in a packet. If an IP header contains additional options other than these, regardless of whether the ASA is configured to allow these options, the ASA will drop the packet.

- **End of Options List (EOOL) or IP Option 0**—This option, which contains just a single zero byte, appears at the end of all options to mark the end of a list of options. This might not coincide with the end of the header according to the header length.

- **No Operation (NOP) or IP Option 1**—The Options field in the IP header can contain zero, one, or more options, which makes the total length of the field variable. However, the IP header must be a multiple of 32 bits. If the number of bits of all options is not a multiple of 32 bits, the NOP option is used as “internal padding” to align the options on a 32-bit boundary.

- **Router Alert (RTRALT) or IP Option 20**—This option notifies transit routers to inspect the contents of the packet even when the packet is not destined for that router. This inspection is valuable when implementing RSVP and similar protocols that require relatively complex processing from the routers along the packet’s delivery path. Dropping RSVP packets containing the Router Alert option can cause problems in VoIP implementations.

**Defaults for IP Options Inspection**

IP Options inspection is enabled by default, using the _default_ip_options_map inspection policy map.

- The Router Alert option is allowed.
- Packets that contain any other options are dropped. This includes packets that contain unsupported options.

Following is the policy map configuration:

```plaintext
policy-map type inspect ip-options _default_ip_options_map
description Default IP-OPTIONS policy-map
parameters
  router-alert action allow
```

**Configure IP Options Inspection**

IP options inspection is enabled by default. You need to configure it only if you want to allow additional options than the default map allows.

**Procedure**

2. Configure the IP Options Inspection Service Policy, page 7-28.
Configure an IP Options Inspection Policy Map

If you want to perform non-default IP options inspection, create an IP options inspection policy map to specify how you want to handle each supported option type.

Procedure

Step 1  Create an IP options inspection policy map:

```plaintext
hostname(config)# policy-map type inspect ip-options policy_map_name
hostname(config-pmap)#
```

Where the `policy_map_name` is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2  (Optional) To add a description to the policy map, enter the following command:

```plaintext
hostname(config-pmap)# description string
```

Step 3  To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```plaintext
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b. Set one or more parameters. You can set the following options; use the `no` form of the command to disable the option. In all cases, the `allow` action allows packets that contain the option without modification; the `clear` action allows the packets but removes the option from the header. Any packet that contains an option that you do not include in the map is dropped. For a description of the options, see Supported IP Options for Inspection, page 7-27.

- **eool** action `{allow | clear}`—Allows or clears the End of Options List option.
- **nop** action `{allow | clear}`—Allows or clears the No Operation option.
- **router-alert** action `{allow | clear}`—Allows or clears the Router Alert (RTRALT) option.

Configure the IP Options Inspection Service Policy

The default ASA configuration includes IP options inspection applied globally on all interfaces. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

Step 1  If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```plaintext
class-map name
match parameter
```

Example:

```plaintext
hostname(config)# class-map ip_options_class_map
hostname(config-cmap)# match access-list ipoptions
```
In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (match default-inspection-traffic). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

Step 2
Add or edit a policy map that sets the actions to take with the class map traffic.

```
policy-map name
```

Example:
```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3
Identify the L3/L4 class map you are using for IP options inspection.

```
class name
```

Example:
```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify inspection_default for the name. Otherwise, you are specifying the class you created earlier in this procedure.

Step 4
Configure IP options inspection.

```
inspect ip-options [ip_options_policy_map]
```

Where ip_options_policy_map is the optional IP options inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the IP options inspection policy map, see Configure an IP Options Inspection Policy Map, page 7-28.

Example:
```
hostname(config-class)# no inspect ip-options
hostname(config-class)# inspect ip-options ip-options-map
```

**Note** If you are editing the default global policy (or any in-use policy) to use a different IP options inspection policy map, you must remove the IP options inspection with the no inspect ip-options command, and then re-add it with the new IP options inspection policy map name.

Step 5
If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

```
service-policy policymap_name {global | interface interface_name}
```

Example:
```
hostname(config)# service-policy global_policy global
```

The global keyword applies the policy map to all interfaces, and interface applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.
Monitoring IP Options Inspection

You can use these techniques to monitor the results of IP options inspection:

- Each time a packet is dropped due to inspection, syslog 106012 is issued. The message shows which option caused the drop.
- Use the `show service-policy inspect ip-options` command to view statistics for each option.

IPsec Pass Through Inspection

The following sections describe the IPsec Pass Through inspection engine.

- IPsec Pass Through Inspection Overview, page 7-30
- Configure IPsec Pass Through Inspection, page 7-30

IPsec Pass Through Inspection Overview

Internet Protocol Security (IPsec) is a protocol suite for securing IP communications by authenticating and encrypting each IP packet of a data stream. IPsec also includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used to protect data flows between a pair of hosts (for example, computer users or servers), between a pair of security gateways (such as routers or firewalls), or between a security gateway and a host.

IPsec Pass Through application inspection provides convenient traversal of ESP (IP protocol 50) and AH (IP protocol 51) traffic associated with an IKE UDP port 500 connection. It avoids lengthy ACL configuration to permit ESP and AH traffic and also provides security using timeout and max connections.

Configure a policy map for IPsec Pass Through to specify the restrictions for ESP or AH traffic. You can set the per client max connections and the idle timeout.

NAT and non-NAT traffic is permitted. However, PAT is not supported.

Configure IPsec Pass Through Inspection

IPsec Pass Through inspection is not enabled by default. You must configure it if you want IPsec Pass Through inspection.

Procedure

- **Step 1** Configure an IPsec Pass Through Inspection Policy Map, page 7-31.
- **Step 2** Configure the IPsec Pass Through Inspection Service Policy, page 7-32.
Configure an IPsec Pass Through Inspection Policy Map

An IPsec Pass Through map lets you change the default configuration values used for IPsec Pass Through application inspection. You can use an IPsec Pass Through map to permit certain flows without using an ACL.

The configuration includes a default map, _default_ipsec_passthru_map, that sets no maximum limit on ESP connections per client, and sets the ESP idle timeout at 10 minutes. You need to configure an inspection policy map only if you want different values, or if you want to set AH values.

Procedure

Step 1 Create an IPsec Pass Through inspection policy map:

```
hostname(config)# policy-map type inspect ipsec-pass-thru policy_map_name
```

Where the policy_map_name is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) To add a description to the policy map, enter the following command:

```
hostname(config-pmap)# description string
```

Step 3 To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```
hostname(config-pmap)# parameters
```

b. Set one or more parameters. You can set the following options; use the no form of the command to disable the option:

- **esp per-client-max number timeout time**—Allows ESP tunnels and sets the maximum connections allowed per client and the idle timeout (in hh:mm:ss format). To allow an unlimited number of connections, specify 0 for the number.

- **ah per-client-max number timeout time**—Allows AH tunnels. The parameters have the same meaning as for the esp command.

Example

The following example shows how to use ACLs to identify IKE traffic, define an IPsec Pass Thru parameter map, define a policy, and apply the policy to the outside interface:

```
hostname(config)# access-list ipsecpassthruacl permit udp any any eq 500
hostname(config)# class-map ipsecpassthru-traffic
hostname(config-cmap)# match access-list ipsecpassthruacl
hostname(config)# policy-map type inspect ipsec-pass-thru iptmap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# esp per-client-max 10 timeout 0:11:00
hostname(config-pmap-p)# ah per-client-max 5 timeout 0:06:00
hostname(config)# policy-map inspection_policy
hostname(config-pmap)# class ipsecpassthru-traffic
hostname(config-pmap-c)# inspect ipsec-pass-thru iptmap
hostname(config)# service-policy inspection_policy interface outside
```
**Configure the IPsec Pass Through Inspection Service Policy**

IPsec Pass Through inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default IPsec ports, so you can simply edit the default global inspection policy to add IPsec inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

**Procedure**

**Step 1**
If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map ipsec_class_map
hostname(config-cmap)# match access-list ipsec
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (match default-inspection-traffic). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

**Step 2**
Add or edit a policy map that sets the actions to take with the class map traffic.

```
policy-map name
```

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

**Step 3**
Identify the L3/L4 class map you are using for IPsec Pass Through inspection.

```
class name
```

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify inspection_default for the name. Otherwise, you are specifying the class you created earlier in this procedure.

**Step 4**
Configure IPsec Pass Through inspection.

```
inspect ipsec-pass-thru [ipsec_policy_map]
```

Where ipsec_policy_map is the optional IPsec Pass Through inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the inspection policy map, see Configure an IPsec Pass Through Inspection Policy Map, page 7-31.

Example:

```
hostname(config-class)# no inspect ipsec-pass-thru
hostname(config-class)# inspect ipsec-pass-thru ipsec-map
```
**Note**

If you are editing the default global policy (or any in-use policy) to use a different IPsec Pass Through inspection policy map, you must remove the IPsec Pass Through inspection with the `no inspect ipsec-pass-thru` command, and then re-add it with the new IPsec Pass Thruough inspection policy map name.

**Step 5**

If you are editing an existing service policy (such as the default global policy called `global_policy`), you are done. Otherwise, activate the policy map on one or more interfaces.

```
service-policy policymap_name {global | interface interface_name}
```

Example:

```
hostname(config)# service-policy global_policy global
```

The `global` keyword applies the policy map to all interfaces, and `interface` applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

---

**IPv6 Inspection**

IPv6 inspection lets you selectively log or drop IPv6 traffic based on the extension header. In addition, IPv6 inspection can check conformance to RFC 2460 for type and order of extension headers in IPv6 packets.

- [Defaults for IPv6 Inspection, page 7-33](#)
- [Configure IPv6 Inspection, page 7-34](#)

**Defaults for IPv6 Inspection**

If you enable IPv6 inspection and do not specify an inspection policy map, then the default IPv6 inspection policy map is used, and the following actions are taken:

- Allows only known IPv6 extension headers. Non-conforming packets are dropped and logged.
- Enforces the order of IPv6 extension headers as defined in the RFC 2460 specification. Non-conforming packets are dropped and logged.
- Drops any packet with a routing type header.

Following is the policy map configuration:

```
policy-map type inspect ipv6 _default_ipv6_map
description Default IPV6 policy-map
parameters
  verify-header type
  verify-header order
match header routing-type range 0 255
drop log
```
Configure IPv6 Inspection

IPv6 inspection is not enabled by default. You must configure it if you want IPv6 inspection.

Procedure

Step 1  Configure an IPv6 Inspection Policy Map, page 7-34.
Step 2  Configure the IPv6 Inspection Service Policy, page 7-35.

Configure an IPv6 Inspection Policy Map

To identify extension headers to drop or log, or to disable packet verification, create an IPv6 inspection policy map to be used by the service policy.

Procedure

Step 1  Create an IPv6 inspection policy map.

```
hostname(config)# policy-map type inspect ipv6 policy_map_name
hostname(config-pmap)#
```

Where the `policy_map_name` is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2  (Optional) Add a description to the policy map.

```
hostname(config-pmap)# description string
```

Step 3  (Optional) Drop or log traffic based on the headers in IPv6 messages.

a. Identify the traffic based on the IPv6 header.

```
hostname(config-pmap)# match header type
```

Where type is one of the following:

- `ah`—Matches the IPv6 Authentication extension header.
- `count gt number`—Specifies the maximum number of IPv6 extension headers, from 0 to 255.
- `destination-option`—Matches the IPv6 destination-option extension header.
- `esp`—Matches the IPv6 Encapsulation Security Payload (ESP) extension header.
- `fragment`—Matches the IPv6 fragment extension header.
- `hop-by-hop`—Matches the IPv6 hop-by-hop extension header.
- `routing-address count gt number`—Sets the maximum number of IPv6 routing header type 0 addresses, greater than a number between 0 and 255.
- `routing-type {eq | range} number`—Matches the IPv6 routing header type, from 0 to 255. For a range, separate values by a space, for example, `30 40`.

b. Specify the action to perform on matching packets. You can drop the packet and optionally log it, or just log it. If you do not enter an action, the packet is logged.

```
hostname(config-pmap)# {drop [log] | log}
```
c. Repeat the process until you identify all headers that you want to drop or log.

Step 4 Configure parameters that affect the inspection engine.

a. Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b. Set one or more parameters. You can set the following options; use the no form of the command to disable the option:

- **verify-header type**—Allows only known IPv6 extension headers.
- **verify-header order**—Enforces the order of IPv6 extension headers as defined in RFC 2460.

---

**Examples**
The following example creates an inspection policy map that will drop and log all IPv6 packets with the hop-by-hop, destination-option, routing-address, and routing type 0 headers. It also enforces header order and type.

```
policy-map type inspect ipv6 ipv6-pm
parameters
verify-header type
verify-header order
match header hop-by-hop
drop log
match header destination-option
drop log
match header routing-address count gt 0
drop log
match header routing-type eq 0
drop log
```

```
policy-map global_policy
class class-default
inspect ipv6 ipv6-pm
!
```

```
service-policy global_policy global
```

---

**Configure the IPv6 Inspection Service Policy**

IPv6 inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. You can simply edit the default global inspection policy to add IPv6 inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

**Procedure**

---

**Step 1** If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map ipv6_class_map
hostname(config-cmap)# match access-list ipv6
```
In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

**Step 2**  
Add or edit a policy map that sets the actions to take with the class map traffic.

```
policy-map name
```

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

**Step 3**  
Identify the L3/L4 class map you are using for IPv6 inspection.

```
class name
```

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify **inspection_default** for the *name*. Otherwise, you are specifying the class you created earlier in this procedure.

**Step 4**  
Configure IPv6 inspection.

```
inspect ipv6 [ipv6_policy_map]
```

Where *ipv6_policy_map* is the optional IPv6 inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the inspection policy map, see Configure an IPv6 Inspection Policy Map, page 7-34.

Example:

```
hostname(config-class)# no inspect ipv6
hostname(config-class)# inspect ipv6 ipv6-map
```

**Note**  
If you are editing the default global policy (or any in-use policy) to use a different IPv6 inspection policy map, you must remove the IPv6 inspection with the **no inspect ipv6** command, and then re-add it with the new IPv6 inspection policy map name.

**Step 5**  
If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

```
service-policy policymap_name {global | interface interface_name}
```

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.
NetBIOS Inspection

NetBIOS inspection is enabled by default. The NetBIOS inspection engine translates IP addresses in the NetBIOS name service (NBNS) packets according to the ASA NAT configuration. You can optionally create a policy map to drop or log NetBIOS protocol violations.

Procedure

Step 1 Configure a NetBIOS Inspection Policy Map for Additional Inspection Control, page 7-37.
Step 2 Configure the NetBIOS Inspection Service Policy, page 7-38.

Configure a NetBIOS Inspection Policy Map for Additional Inspection Control

To specify the action for protocol violations, create a NETBIOS inspection policy map. You can then apply the inspection policy map when you enable NETBIOS inspection.

Procedure

Step 1 Create a NetBIOS inspection policy map.
hostname(config)# policy-map type inspect netbios policy_map_name
hostname(config-pmap)#

Where the policy_map_name is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) To add a description to the policy map, enter the following command:
hostname(config-pmap)# description string

Step 3 Enter parameters configuration mode.
hostname(config-pmap)# parameters
hostname(config-pmap-p)#

Step 4 Specify the action to take for NETBIOS protocol violations.
hostname(config-pmap-p)# protocol-violation action {drop [log] | log}

Where the drop action drops the packet. The log action sends a system log message when this policy map matches traffic.

Example
hostname(config)# policy-map type inspect netbios netbios_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# protocol-violation drop log

hostname(config)# policy-map netbios_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect netbios netbios_map
Configure the NetBIOS Inspection Service Policy

NetBIOS application inspection performs NAT for the embedded IP address in the NetBIOS name service packets and NetBIOS datagram services packets. It also enforces protocol conformance, checking the various count and length fields for consistency.

The default ASA configuration includes NetBIOS inspection on the default port applied globally on all interfaces. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

**Step 1** If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map netbios_class_map
hostname(config-cmap)# match access-list netbios
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Identify Traffic (Layer 3/4 Class Maps), page 1-13.

**Step 2** Add or edit a policy map that sets the actions to take with the class map traffic.

```
policy-map name
```

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

**Step 3** Identify the L3/L4 class map you are using for NetBIOS inspection.

```
class name
```

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify **inspection_default** for the **name**. Otherwise, you are specifying the class you created earlier in this procedure.

**Step 4** Configure NetBIOS inspection.

```
inspect netbios [netbios_policy_map]
```

Where **netbios_policy_map** is the optional NetBIOS inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the NetBIOS inspection policy map, see Configure a NetBIOS Inspection Policy Map for Additional Inspection Control, page 7-37.

Example:

```
hostname(config-class)# no inspect netbios
```
hostname(config-class)# inspect netbios netbios-map

Note
If you are editing the default global policy (or any in-use policy) to use a different NetBIOS inspection policy map, you must remove the NetBIOS inspection with the no inspect skinny command, and then re-add it with the new NetBIOS inspection policy map name.

Step 5
If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy policymap_name {global | interface interface_name}

Example:
hostname(config)# service-policy global_policy global

The global keyword applies the policy map to all interfaces, and interface applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

PPTP Inspection

PPTP is a protocol for tunneling PPP traffic. A PPTP session is composed of one TCP channel and usually two PPTP GRE tunnels. The TCP channel is the control channel used for negotiating and managing the PPTP GRE tunnels. The GRE tunnels carry PPP sessions between the two hosts.

When enabled, PPTP application inspection inspects PPTP protocol packets and dynamically creates the GRE connections and xlates necessary to permit PPTP traffic.

Specifically, the ASA inspects the PPTP version announcements and the outgoing call request/response sequence. Only PPTP Version 1, as defined in RFC 2637, is inspected. Further inspection on the TCP control channel is disabled if the version announced by either side is not Version 1. In addition, the outgoing-call request and reply sequence are tracked. Connections and xlates are dynamic allocated as necessary to permit subsequent secondary GRE data traffic.

The PPTP inspection engine must be enabled for PPTP traffic to be translated by PAT. Additionally, PAT is only performed for a modified version of GRE (RFC2637) and only if it is negotiated over the PPTP TCP control channel. PAT is not performed for the unmodified version of GRE (RFC 1701 and RFC 1702).

For information on enabling PPTP inspection, see Configure Application Layer Protocol Inspection, page 6-9.

SMTP and Extended SMTP Inspection

ESMTP inspection detects attacks, including spam, phising, malformed message attacks, buffer overflow/underflow attacks. It also provides support for application security and protocol conformance, which enforce the sanity of the ESMTP messages as well as detect several attacks, block senders/receivers, and block mail relay.

The following sections describe the ESMTP inspection engine.

- SMTP and ESMTP Inspection Overview, page 7-40
SMTP and ESMTP Inspection Overview

ESMTP application inspection provides improved protection against SMTP-based attacks by restricting the types of SMTP commands that can pass through the ASA and by adding monitoring capabilities.

ESMTP is an enhancement to the SMTP protocol and is similar in most respects to SMTP. For convenience, the term SMTP is used in this document to refer to both SMTP and ESMTP. The application inspection process for extended SMTP is similar to SMTP application inspection and includes support for SMTP sessions. Most commands used in an extended SMTP session are the same as those used in an SMTP session but an ESMTP session is considerably faster and offers more options related to reliability and security, such as delivery status notification.

Extended SMTP application inspection adds support for these extended SMTP commands, including AUTH, EHLO, ETRN, HELP, SAML, SEND, SOML, STARTTLS, and VRFY. Along with the support for seven RFC 821 commands (DATA, HELO, MAIL, NOOP, QUIT, RCPT, RSET), the ASA supports a total of fifteen SMTP commands.

Other extended SMTP commands, such as ATRN, ONEX, VERB, CHUNKING, and private extensions are not supported. Unsupported commands are translated into Xs, which are rejected by the internal server. This results in a message such as “500 Command unknown: ’XXX’.” Incomplete commands are discarded.

The ESMTP inspection engine changes the characters in the server SMTP banner to asterisks except for the “2”, “0”, “0” characters. Carriage return (CR) and linefeed (LF) characters are ignored.

With SMTP inspection enabled, a Telnet session used for interactive SMTP may hang if the following rules are not observed: SMTP commands must be at least four characters in length; must be terminated with carriage return and line feed; and must wait for a response before issuing the next reply.

An SMTP server responds to client requests with numeric reply codes and optional human-readable strings. SMTP application inspection controls and reduces the commands that the user can use as well as the messages that the server returns. SMTP inspection performs three primary tasks:

- Restricts SMTP requests to seven basic SMTP commands and eight extended commands.
- Monitors the SMTP command-response sequence.
- Generates an audit trail—Audit record 108002 is generated when an invalid character embedded in the mail address is replaced. For more information, see RFC 821.

SMTP inspection monitors the command and response sequence for the following anomalous signatures:

- Truncated commands.
- Incorrect command termination (not terminated with <CR><LR>).
- The MAIL and RCPT commands specify who are the sender and the receiver of the mail. Mail addresses are scanned for strange characters. The pipeline character (|) is deleted (changed to a blank space) and “<” “>” are only allowed if they are used to define a mail address (“>” must be preceded by “<”).
- Unexpected transition by the SMTP server.
- For unknown commands, the ASA changes all the characters in the packet to X. In this case, the server generates an error code to the client. Because of the change in the packet, the TCP checksum has to be recalculated or adjusted.
- TCP stream editing.

- Defaults for ESMTP Inspection, page 7-41
- Configure ESMTP Inspection, page 7-42
Chapter 7  Inspection of Basic Internet Protocols

SMTP and Extended SMTP Inspection

- Command pipelining.

### Defaults for ESMTP Inspection

ESMTP inspection is enabled by default, using the `_default_esmtp_map` inspection policy map.

- The server banner is masked.
- Encrypted connections are not allowed. The STARTTLS indication is removed from the session connection attempt, forcing the client and server to negotiate a plain text session, which can be inspected.
- Special characters in sender and receiver address are not noticed, no action is taken.
- Connections with command line length greater than 512 are dropped and logged.
- Connections with more than 100 recipients are dropped and logged.
- Messages with body length greater than 998 bytes are logged.
- Connections with header line length greater than 998 are dropped and logged.
- Messages with MIME filenames greater than 255 characters are dropped and logged.
- EHLO reply parameters matching “others” are masked.

Following is the policy map configuration:

```plaintext
policy-map type inspect esmtp _default_esmtp_map
description Default ESMTP policy-map
parameters
    mask-banner
    no mail-relay
    no special-character
    no allow-tls
match cmd line length gt 512
drop-connection log
match cmd RCPT count gt 100
drop-connection log
match body line length gt 998
log
match header line length gt 998
drop-connection log
match sender-address length gt 320
drop-connection log
match MIME filename length gt 255
drop-connection log
match ehlo-reply-parameter others
    mask
```
Configure ESMTP Inspection

ESMTP inspection is enabled by default. You need to configure it only if you want to different process than that provided by the default inspection map.

Procedure

Step 1 Configure an ESMTP Inspection Policy Map, page 7-42.
Step 2 Configure the ESMTP Inspection Service Policy, page 7-44.

Configure an ESMTP Inspection Policy Map

To specify actions when a message violates a parameter, create an ESMTP inspection policy map. You can then apply the inspection policy map when you enable ESMTP inspection.

Before You Begin
Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 Create an ESMTP inspection policy map, enter the following command:

hostname(config)# policy-map type inspect esmtp policy_map_name
hostname(config-pmap)#

Where the policy_map_name is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) To add a description to the policy map, enter the following command:

hostname(config-pmap)# description string

Step 3 To apply actions to matching traffic, perform the following steps.

a. Specify the traffic on which you want to perform actions using one of the following match commands. If you use a match not command, then any traffic that does not match the criterion in the match not command has the action applied.

- match [not] body {length | line length} gt bytes—Matches messages where the length or length of a line in an ESMTP body message is greater than the specified number of bytes.
- match [not] cmd verb verb1 [verb2...]—Matches the command verb in the message. You can specify one or more of the following commands: auth, data, ehlo, etrn, helo, help, mail, noop, quit, rcpt, rset, saml, soml, vrfy.
- match [not] cmd line length gt bytes—Matches messages where the length of a line in the command verb is greater than the specified number of bytes.
- match [not] cmd rcpt count gt count—Matches messages where the number of recipients is greater than the specified count.
- match [not] ehlo-reply-parameter parameter [parameter2...]—Matches ESMTP EHLO reply parameters. You can specify one or more of the following parameters: 8bitmime, auth, binaryname, checkpoint, dsn, etrn, others, pipelining, size, vrfy.
**match [not] header {length | line length} gt bytes**—Matches messages where the length or length of a line in an ESMTP header is greater than the specified number of bytes.

**match [not] header to-fields count gt count**—Matches messages where the number of To fields in the header is greater than the specified number.

**match [not] invalid-recipients count gt number**—Matches messages where the number of invalid recipients is greater than the specified count.

**match [not] mime filetype regex {regex_name | class class_name}**—Matches the MIME or media file type against the specified regular expression or regular expression class.

**match [not] mime filename length gt bytes**—Matches messages where a file name is longer than the specified number of bytes.

**match [not] mime encoding type [type2...]**—Matches the MIME encoding type. You can specify one or more of the following types: 7bit, 8bit, base64, binary, others, quoted-printable.

**match [not] sender-address regex {regex_name | class class_name}**—Matches the sender email address against the specified regular expression or regular expression class.

**match [not] sender-address length gt bytes**—Matches messages where the sender address is greater than the specified number of bytes.

b. Specify the action you want to perform on the matching traffic by entering the following command:

```plaintext
hostname(config-pmap-c)# {drop-connection [log] | mask [log] | reset [log] | log | rate-limit message_rate}
```

Not all options are available for each `match` command. See the CLI help or the command reference for the exact options available.

- The `drop-connection` keyword drops the packet and closes the connection.
- The `mask` keyword masks out the matching portion of the packet. This action is available for `ehlo-reply-parameter` and `cmd verb` only.
- The `reset` keyword drops the packet, closes the connection, and sends a TCP reset to the server and/or client.
- The `log` keyword, which you can use alone or with one of the other keywords, sends a system log message.
- The `rate-limit message_rate` argument limits the rate of messages. This option is available with `cmd verb` only, where you can use it as the only action, or you can use it in conjunction with the `mask` action.

You can specify multiple `match` commands in the policy map. For information about the order of `match` commands, see [Defining Actions in an Inspection Policy Map](#), page 2-4.

**Step 4** To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```plaintext
hostname(config-pmap-c)# parameters
hostname(config-pmap-c-p)#
```

b. Set one or more parameters. You can set the following options; use the `no` form of the command to disable the option:

- **mail-relay domain-name action {drop-connection [log] | log]**—Identifies a domain name for mail relay. You can either drop the connection and optionally log it, or log it.
- **mask-banner**— Masks the banner from the ESMTP server.
- **special-character action** \{drop-connection [log] | log\}—Identifies the action to take for messages that include the special characters pipe (|), back quote, and NUL in the sender or receiver email addresses. You can either drop the connection and optionally log it, or log it.

- **allow-tls** [action log]—Whether to allow ESMTP over TLS (encrypted connections) without inspection. You can optionally log encrypted connections. The default is **no allow-tls**, which strips the STARTTLS indication from the session connection and forces a plain-text connection.

### Example
The following example shows how to define an ESMTP inspection policy map.

```
hostname(config)# regex user1 "user1@cisco.com"
hostname(config)# regex user2 "user2@cisco.com"
hostname(config)# regex user3 "user3@cisco.com"
hostname(config)# class-map type regex senders_black_list
hostname(config-cmap)# description "Regular expressions to filter out undesired senders"
hostname(config-cmap)# match regex user1
hostname(config-cmap)# match regex user2
hostname(config-cmap)# match regex user3

hostname(config)# policy-map type inspect esmtp advanced_esmtp_map
hostname(config-pmap)# match sender-address regex class senders_black_list
hostname(config-pmap-c)# drop-connection log

hostname(config)# policy-map outside_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect esmtp advanced_esmtp_map

hostname(config)# service-policy outside_policy interface outside
```

### Configure the ESMTP Inspection Service Policy
The default ASA configuration includes ESMTP inspection applied globally on all interfaces. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

#### Procedure

**Step 1** If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map esmtp_class_map
hostname(config-cmap)# match access-list esmtp
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see **Identify Traffic (Layer 3/4 Class Maps)**, page 1-13.

**Step 2** Add or edit a policy map that sets the actions to take with the class map traffic.

```
policy-map name
```
Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the `global_policy` policy map is assigned globally to all interfaces. If you want to edit the `global_policy`, enter `global_policy` as the policy name.

**Step 3** Identify the L3/L4 class map you are using for IP options inspection.

```
class name
```

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special `inspection_default` class map in a new policy, specify `inspection_default` for the `name`. Otherwise, you are specifying the class you created earlier in this procedure.

**Step 4** Configure ESMTP inspection.

```
inspect esmtp [esmtp_policy_map]
```

Where `esmtp_policy_map` is the optional ESMTP inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the ESMTP inspection policy map, see Configure the ESMTP Inspection Service Policy, page 7-44.

Example:

```
hostname(config-class)# no inspect esmtp
hostname(config-class)# inspect esmtp esmtp-map
```

---

**Note** If you are editing the default global policy (or any in-use policy) to use a different inspection policy map, you must remove the ESMTP inspection with the `no inspect esmtp` command, and then re-add it with the new inspection policy map name.

**Step 5** If you are editing an existing service policy (such as the default global policy called `global_policy`), you are done. Otherwise, activate the policy map on one or more interfaces.

```
service-policy policy_name map (global | interface interface_name)
```

Example:

```
hostname(config)# service-policy global_policy global
```

The `global` keyword applies the policy map to all interfaces, and `interface` applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

---

**TFTP Inspection**

TFTP inspection is enabled by default.

TFTP, described in RFC 1350, is a simple protocol to read and write files between a TFTP server and client.
The ASA inspects TFTP traffic and dynamically creates connections and translations, if necessary, to permit file transfer between a TFTP client and server. Specifically, the inspection engine inspects TFTP read request (RRQ), write request (WRQ), and error notification (ERROR).

A dynamic secondary channel and a PAT translation, if necessary, are allocated on a reception of a valid read (RRQ) or write (WRQ) request. This secondary channel is subsequently used by TFTP for file transfer or error notification.

Only the TFTP server can initiate traffic over the secondary channel, and at most one incomplete secondary channel can exist between the TFTP client and server. An error notification from the server closes the secondary channel.

TFTP inspection must be enabled if static PAT is used to redirect TFTP traffic.

For information on enabling TFTP inspection, see Configure Application Layer Protocol Inspection, page 6-9.