Configuring Special Actions for Application Inspections (Inspection Policy Map)

Modular Policy Framework lets you configure special actions for many application inspections. When you enable an inspection engine in the Layer 3/4 policy map, you can also optionally enable actions as defined in an *inspection policy map*. When the inspection policy map matches traffic within the Layer 3/4 class map for which you have defined an inspection action, then that subset of traffic will be acted upon as specified (for example, dropped or rate-limited).

This chapter includes the following sections:

- Information About Inspection Policy Maps, page 2-1
- Guidelines and Limitations, page 2-2
- Default Inspection Policy Maps, page 2-3
- Defining Actions in an Inspection Policy Map, page 2-4
- Identifying Traffic in an Inspection Class Map, page 2-5
- Where to Go Next, page 2-7
- Feature History for Inspection Policy Maps, page 2-7

Information About Inspection Policy Maps

See the "Configuring Application Layer Protocol Inspection" section on page 9-7 for a list of applications that support inspection policy maps.

An inspection policy map consists of one or more of the following elements. The exact options available for an inspection policy map depends on the application.

- Traffic matching command—You can define a traffic matching command directly in the inspection policy map to match application traffic to criteria specific to the application, such as a URL string, for which you then enable actions.
  - Some traffic matching commands can specify regular expressions to match text inside a packet. Be sure to create and test the regular expressions before you configure the policy map, either singly or grouped together in a regular expression class map.
- Inspection class map—An inspection class map includes multiple traffic matching commands. You then identify the class map in the policy map and enable actions for the class map as a whole. The difference between creating a class map and defining the traffic match directly in the inspection

policy map is that you can create more complex match criteria and you can reuse class maps. However, you cannot set different actions for different matches. **Note:** Not all inspections support inspection class maps.

- Parameters—Parameters affect the behavior of the inspection engine.

Guidelines and Limitations

- HTTP inspection policy maps—If you modify an in-use HTTP inspection policy map (**policy-map type inspect http**), you must remove and reapply the **inspect http** *map* action for the changes to take effect. For example, if you modify the "http-map" inspection policy map, you must remove and readd the **inspect http http-map** command from the layer 3/4 policy:

```
ciscoasa(config)# policy-map test
ciscoasa(config-pmap)# class http
ciscoasa(config-pmap-c)# no inspect http http-map
ciscoasa(config-pmap-c)# inspect http http-map
```

- All inspection policy maps—If you want to exchange an in-use inspection policy map for a different map name, you must remove the **inspect** *protocol map* command, and readd it with the new map. For example:

```
ciscoasa(config)# policy-map test
ciscoasa(config-pmap)# class sip
ciscoasa(config-pmap-c)# no inspect sip sip-map1
ciscoasa(config-pmap-c)# inspect sip sip-map2
```

- You can specify multiple **class** or **match** commands in the inspection policy map.

  If a packet matches multiple different **match** or **class** commands, then the order in which the ASA applies the actions is determined by internal ASA rules, and not by the order they are added to the inspection policy map. The internal rules are determined by the application type and the logical progression of parsing a packet, and are not user-configurable. For example for HTTP traffic, parsing a Request Method field precedes parsing the Header Host Length field; an action for the Request Method field occurs before the action for the Header Host Length field. For example, the following match commands can be entered in any order, but the **match request method get** command is matched first.

```
match request header host length gt 100
  reset
match request method get
  log
```

  If an action drops a packet, then no further actions are performed in the inspection policy map. For example, if the first action is to reset the connection, then it will never match any further **match** or **class** commands. If the first action is to log the packet, then a second action, such as resetting the connection, can occur.

  If a packet matches multiple **match** or **class** commands that are the same, then they are matched in the order they appear in the policy map. For example, for a packet with the header length of 1001, it will match the first command below, and be logged, and then will match the second command and be reset. If you reverse the order of the two **match** commands, then the packet will be dropped and the connection reset before it can match the second **match** command; it will never be logged.

```
match request header length gt 100
  log
match request header length gt 1000
  reset
```

A class map is determined to be the same type as another class map or **match** command based on the lowest priority **match** command in the class map (the priority is based on the internal rules). If a class map has the same type of lowest priority **match** command as another class map, then the class maps are matched according to the order they are added to the policy map. If the lowest priority match for each class map is different, then the class map with the higher priority **match** command is matched first. For example, the following three class maps contain two types of **match** commands: **match request-cmd** (higher priority) and **match filename** (lower priority). The ftp3 class map includes both commands, but it is ranked according to the lowest priority command, **match filename**. The ftp1 class map includes the highest priority command, so it is matched first, regardless of the order in the policy map. The ftp3 class map is ranked as being of the same priority as the ftp2 class map, which also contains the **match filename** command. They are matched according to the order in the policy map: ftp3 and then ftp2.

```
class-map type inspect ftp match-all ftp1
  match request-cmd get
class-map type inspect ftp match-all ftp2
  match filename regex abc
class-map type inspect ftp match-all ftp3
  match request-cmd get
  match filename regex abc

policy-map type inspect ftp ftp
  class ftp3
    log
  class ftp2
    log
  class ftp1
    log
```

Default Inspection Policy Maps

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 commands:

```
policy-map type inspect dns preset_dns_map
 parameters
    message-length maximum client auto
    message-length maximum 512
    dns-guard
    protocol-enforcement
    nat-rewrite
```

Note There are other default inspection policy maps such as **_default_esmtp_map**. For example, **inspect esmtp** implicitly uses the policy map "_default_esmtp_map." All the default policy maps can be shown by using the **show running-config all policy-map** command.

Defining Actions in an Inspection Policy Map

When you enable an inspection engine in the Layer 3/4 policy map, you can also optionally enable actions as defined in an inspection policy map.

Detailed Steps

	Command	Purpose
Step 1	(Optional) Create an inspection class map.	See the "Identifying Traffic in an Inspection Class Map" section on page 2-5. Alternatively, you can identify the traffic directly within the policy map.
Step 2	(Optional) Create a regular expression.	For policy map types that support regular expressions, see the general operations configuration guide.
Step 3	**policy-map type inspect** *application policy_map_name* **Example:** `ciscoasa(config)# policy-map type inspect http http_policy`	Creates the inspection policy map. See the "Configuring Application Layer Protocol Inspection" section on page 9-7 for a list of applications that support inspection policy maps. The *policy_map_name* argument is the name of the policy map up to 40 characters in length. All types of policy maps use the same name space, so you cannot reuse a name already used by another type of policy map. The CLI enters policy-map configuration mode.
Step 4	Specify the traffic on which you want to perform actions using one of the following methods:	

	class *class_map_name*
	Example:
	`ciscoasa(config-pmap)# class http_traffic`
	`ciscoasa(config-pmap-c)#`
		Specifies the inspection class map that you created in the "Identifying Traffic in an Inspection Class Map" section on page 2-5.
		Not all applications support inspection class maps.

	Specify traffic directly in the policy map using one of the **match** commands described for each application in the inspection chapter.
	Example:
	`ciscoasa(config-pmap)# match req-resp content-type mismatch`
	`ciscoasa(config-pmap-c)#`
		If you use a **match not** command, then any traffic that matches the criterion in the **match not** command does not have the action applied.
		For policy map types that support regular expressions, see the general operations configuration guide.

	Command	Purpose
Step 5	*action* **Example:** `ciscoasa(config-pmap-c)# drop-connection log`	Specifies the action you want to perform on the matching traffic. Actions vary depending on the inspection and match type. Common actions include: **drop**, **log**, and **drop-connection**. For the actions available for each match, see the appropriate inspection chapter.

Step 6	**parameters**
	
	Example:
	`ciscoasa(config-pmap)# parameters`
	`ciscoasa(config-pmap-p)#`
		Configures parameters that affect the inspection engine. The CLI enters parameters configuration mode. For the parameters available for each application, see the appropriate inspection chapter.

Examples

The following is an example of an HTTP inspection policy map and the related class maps. This policy map is activated by the Layer 3/4 policy map, which is enabled by the service policy.

```
ciscoasa(config)# regex url_example example\.com
ciscoasa(config)# regex url_example2 example2\.com
ciscoasa(config)# class-map type regex match-any URLs
ciscoasa(config-cmap)# match regex url_example
ciscoasa(config-cmap)# match regex url_example2

ciscoasa(config-cmap)# class-map type inspect http match-all http-traffic
ciscoasa(config-cmap)# match req-resp content-type mismatch
ciscoasa(config-cmap)# match request body length gt 1000
ciscoasa(config-cmap)# match not request uri regex class URLs

ciscoasa(config-cmap)# policy-map type inspect http http-map1
ciscoasa(config-pmap)# class http-traffic
ciscoasa(config-pmap-c)# drop-connection log
ciscoasa(config-pmap-c)# match req-resp content-type mismatch
ciscoasa(config-pmap-c)# reset log
ciscoasa(config-pmap-c)# parameters
ciscoasa(config-pmap-p)# protocol-violation action log

ciscoasa(config-pmap-p)# policy-map test
ciscoasa(config-pmap)# class test (a Layer 3/4 class map not shown)
ciscoasa(config-pmap-c)# inspect http http-map1

ciscoasa(config-pmap-c)# service-policy test interface outside
```

Identifying Traffic in an Inspection Class Map

This type of class map allows you to match criteria that is specific to an application. For example, for DNS traffic, you can match the domain name in a DNS query.

A class map groups multiple traffic matches (in a match-all class map), or lets you match any of a list of matches (in a match-any class 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 group multiple match commands, and you can reuse class maps. For the traffic that you identify in this class map, you can specify actions such as dropping, resetting, and/or logging the connection in the inspection policy map. If you want to perform different actions on different types of traffic, you should identify the traffic directly in the policy map.

Restrictions

Not all applications support inspection class maps. See the CLI help for **class-map type inspect** for a list of supported applications.

Detailed Steps

	Command	Purpose
Step 1	(Optional) Create a regular expression.	See the general operations configuration guide.

Step 2	**class-map type inspect** *application* [**match-all** | **match-any**] *class_map_name*
	
	Example:
	`ciscoasa(config)# class-map type inspect http http_traffic`
	`ciscoasa(config-cmap)#`
		Creates an inspection class map, where the *application* is the application you want to inspect. For supported applications, see the CLI help for a list of supported applications or see Chapter 9, "Getting Started with Application Layer Protocol Inspection."
		
		The *class_map_name* argument is the name of the class map up to 40 characters in length.
		
		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 of the criteria.
		
		The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

Step 3	(Optional) **description** *string* **Example:** `hostname(config-cmap)# description All UDP traffic`	Adds a description to the class map.
Step 4	Define the traffic to include in the class by entering one or more **match** commands available for your application.	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. To see the **match** commands available for each application, see the appropriate inspection chapter.

Examples

The following example creates an HTTP class map that must match all criteria:

```
ciscoasa(config-cmap)# class-map type inspect http match-all http-traffic
ciscoasa(config-cmap)# match req-resp content-type mismatch
ciscoasa(config-cmap)# match request body length gt 1000
ciscoasa(config-cmap)# match not request uri regex class URLs
```

The following example creates an HTTP class map that can match any of the criteria:

```
ciscoasa(config-cmap)# class-map type inspect http match-any monitor-http
ciscoasa(config-cmap)# match request method get
ciscoasa(config-cmap)# match request method put
ciscoasa(config-cmap)# match request method post
```

Where to Go Next

To use an inspection policy, see Chapter 1, "Configuring a Service Policy Using the Modular Policy Framework."

Feature History for Inspection Policy Maps

Table 2-1 lists the release history for this feature.

Table 2-1 Feature History for Service Policies

Feature Name	Releases	Feature Information
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following command was introduced: **class-map type inspect**.
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Match any for inspection policy maps	8.0(2)	The **match any** keyword was introduced for use with inspection policy maps: traffic can match one or more criteria to match the class map. Formerly, only **match all** was available.