CHAPTER 31

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

See the "Configuring Application Layer Protocol Inspection" section on page 38-6 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—(Not available for all applications. See the CLI help for a list of supported applications.) An inspection class map includes traffic matching commands that match application traffic with criteria specific to the application, such as a URL string. You then identify the class map in the policy map and enable actions. 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.

  - 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.
- 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:

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

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

Default Inspection Policy Maps

The default inspection policy map configuration includes the following commands, which sets the maximum message length for DNS packets to be 512 bytes:

```
policy-map type inspect dns preset_dns_map
 parameters
  message-length maximum 512
```

Note There are other default inspection policy maps such as **policy-map type inspect esmtp _default_esmtp_map**. These default policy maps are created implicitly by the command **inspect** *protocol*. 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.

Restrictions

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

If a packet matches multiple different **match** or **class** commands, then the order in which the adaptive security appliance applies the actions is determined by internal adaptive security appliance rules, and not by the order they are added to the 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. (You can configure both the **reset** (or **drop-connection**, and so on.) and the **log** action for the same **match** or **class** command, in which case the packet is logged before it is reset for a given match.)

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 command 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
```

Detailed Steps

Step 1 (Optional) Create an inspection class map according to the "Identifying Traffic in an Inspection Class Map" section on page 31-5. Alternatively, you can identify the traffic directly within the policy map.

Step 2 To create the inspection policy map, enter the following command:

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

See the "Configuring Application Layer Protocol Inspection" section on page 38-6 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 3 To apply actions to matching traffic, perform the following steps.

Note For information about including multiple **class** or **match** commands, see the "Restrictions" section on page 31-2.

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

- Specify the inspection class map that you created in the "Identifying Traffic in an Inspection Class Map" section on page 31-5 by entering the following command:

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

  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 applicable inspection chapter. 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.

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

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

Not all options are available for each application. Other actions specific to the application might also be available. See the appropriate inspection chapter for the exact options available.

The **drop** keyword drops all packets that match.

The **send-protocol-error** keyword sends a protocol error message.

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

The **mask** keyword masks out the matching portion of the packet.

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.

Step 4 To configure parameters that affect the inspection engine, enter the following command:

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

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.

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

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

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

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

hostname(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

Step 1 (Optional) If you want to match based on a regular expression, see the "Creating a Regular Expression" section on page 11-12 and the "Creating a Regular Expression Class Map" section on page 11-15.

Step 2 Create a class map by entering the following command:

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

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 38, "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) To add a description to the class map, enter the following command:

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

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:

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

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

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

Where to Go Next

To use an inspection policy, see Chapter 30, "Configuring a Service Policy Using the Modular Policy Framework."