Flexible Packet Matching

Last Updated: January 19, 2012

Flexible Packet Matching (FPM) is an access control list (ACL) pattern matching tool, providing more thorough and customized packet filters. FPM enables users to match on arbitrary bits of a packet at an arbitrary depth in the packet header and payload. FPM removes constraints to specific fields that had limited packet inspection.

FPM enables users to create their own stateless packet classification criteria and to define policies with multiple actions (such as drop, log, or send Internet Control Message Protocol [ICMP] unreachable[1]) to immediately block new viruses, worms, and attacks.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Flexible Packet Matching

- In Cisco IOS Release 12.4(4)T, FPM is available only in advanced security images.

1 Send ICMP unreachable is currently not supported on the Supervisor Engine 32 PISA.

- In Cisco IOS Release 12.2(18)ZY, FPM is available in ipbase and ipservices images for the Supervisor Engine 32 Programmable Intelligent Services Accelerator (PISA) platform.
- Although access to an XML editor is not required, XML will ease the creation of protocol header description files (PHDFs).
- FPM cannot be used to mitigate an attack that requires stateful classification.
- Because FPM is stateless, it cannot keep track of port numbers being used by protocols that dynamically negotiate ports. Thus port numbers must be explicitly specified when using FPM.
- FPM cannot perform IP fragmentation or TCP flow reassembly.
- FPM inspects only IPv4 unicast packets.
- FPM cannot classify packets with IP options.
- FPM does not support multicast packet inspection.
- FPM is not supported on tunnel and Multiprotocol Label Switching (MPLS) interfaces.
- FPM cannot be configured on FlexWAN cards.
- Noninitial fragments will not be matched by the FPM engine.
- Offset can be a constant only in a match start construct.
- FPM cannot match across packets.
- Mapping of FPM policies to the control plane is not supported.

Information About Flexible Packet Matching

Flexible Packet Matching Functional Overview

FPM allows customers to create their own filtering policies that can immediately detect and block new viruses and attacks.

A filtering policy is defined via the following tasks:

- Load a PHDF (for protocol header field matching)
- Define a class map and define the protocol stack chain (traffic class)
- Define a service policy (traffic policy)
- Apply the service policy to an interface

Protocol Header Description File

Protocol headers are defined in separate files called PHDFs; the field names that are defined within the PHDFs are used for defining the packet filters. A PHDF is a file that allows the user to leverage the

flexibility of XML to describe almost any protocol header. The important components of the PHDF are the version, the XML file schema location, and the protocol field definitions. The protocol field definitions name the appropriate field in the protocol header, allow for a comment describing the field, provide the location of the protocol header field in the header (the offset is relative to the start of the protocol header), and provide the length of the field. Users can choose to specify the measurement in bytes or in bits.

Note The total length of the header must be specified at the end of each PHDF.

Note When redundant sup PHDF files are used by the FPM policy, the files should also be on the standby sup's corresponding disk. If the files are not available the FPM policy will not work after the switchover.

Users can write their own custom PHDFs via XML for existing or proprietary protocols. However, the following standard PHDFs can also be loaded onto the router via the **load protocol** command: ether.phdf, ip.phdf, tcp.phdf, and udp.phdf.

Note Because PHDFs are defined via XML, they are not shown in a running configuration. However, you can use the **show protocol phdf** command to verify the loaded PHDF.

Standard PHDFs are available on Cisco.com at the following URL: http://www.cisco.com/cgi-bin/tablebuild.pl/fpm

Filter Description

A filter description is a definition of a traffic class that can contain the header fields defined in a PHDF (using the **match field** command). If a PHDF is not loaded, the traffic class can be defined through the datagram header start (Layer 2) or the network header start (Layer 3) (using the **match start** command). If a PHDF has been loaded onto the router, the class specification begins with a list of the protocol headers in the packet.

A filter definition also includes the policy map; that is, after a class map has been defined, a policy map is needed to bind the match to an action. A policy map is an ordered set of classes and associated actions, such as drop, log, or send ICMP unreachable.

For information on how to configure a class map and a policy map for FPM, see the How to Configure a Flexible Packet Matching Traffic Class and Traffic Policy section.

Traffic Classification Definition Files for the Flexible Packet Matching XML Configuration

FPM uses a traffic classification definition file (TCDF) to define policies that can block attacks on the network. Before Cisco IOS Release 12.4(6)T, FPM defined traffic classes (class maps), policies (policy maps), and service policies (attach policy maps to class maps) through the use of the command line interface (CLI). With TCDFs, FPM can use XML as an alternative to the CLI to define classes of traffic and specify actions to apply to the traffic classes. Traffic classification behavior is the same whether you create the behavior using a TCDF or configure it using CLI commands. Once a TCDF is created, it can be loaded on any FPM-enabled device in the network.

Note TCDFs are supported only in Cisco IOS Release 12.4(6)T and later T-train releases.

For more information on configuring FPM using TCDFs, see "Flexible Packet Matching XML Configuration".

FPM on the Catalyst 6500 Equipped with PISA Overview

The PISA functions as a network processor-based daughter card that is mounted on the Catalyst 6500 Supervisor. PISA provides a superset of the multilater switch feature card 2a (MSFC2a) capabilities. In addition to performing all of the same functions as the MSFC2a, PISA provides dedicated hardware to accelerate certain features such as FPM.

Network-Based Application Recognition (NBAR) occurs before FPM; thus, packets that are dropped by FPM are processed by NBAR.

Logging FPM Activity

In software-based FPM logging, every flow is logged and aggregated statistics are provided for each flow. Logging every flow for FPM on PISA would overwhelm the CPU; thus, only selective packets are logged. That is, when a packet matches a policy that is to be logged or the first time, the packet is logged, time-stamped, and stored. For every subsequent packet that matches any policy with a log action, the packet is checked for the difference between the current time (which is clocked by the global timer) and the last time stamp. If the current time is later than the last time stamp, the packet is logged and the “stamp time” is updated with the current time.

Memory Requirements

Note Because memory requirements vary among system configurations, the requirements listed in this document are estimates.

- PISA will support a maximum of 1024 interfaces; however, it is expected that no more than 256 interfaces will be configured with FPM.
- A maximum of 32 classes per policy map, and a total of 1024 classes globally, are supported.
- A maximum of 32 filters (such as match entries) per class map are supported. (However, some optimizations for better performance are possible with match-any type of class maps that have filters starting at the same offset and the same size.)

Encrypted TCDF Support

TCDFs provide preconfigured FPM filters written in XML format that can be directly loaded onto a router. The XML format prohibits the Cisco Product Security Incident Response Team (PSIRT) from being able to provide public TCDF filters because it would expose the vulnerability to potential attackers. This information could then be used to exploit PSIRT vulnerabilities in some systems.

FPM encrypted TCDF (eTCDF) filter support will provide encrypted FPM filters. Applying the PSIRT provided eTCDF FPM filter will protect routers from PSIRT incidents, allowing time to certify new Cisco IOS releases that contains the PSIRT fixes.

To enter FPM match encryption filter configuration mode, use the **match encrypted** command in class-map configuration mode. This mode enables you to enter encrypted filter-related information like the cipher key cipher value, and filter hash.

Note The encrypted filter contents are not stored in the class map until the **exit** or **end**commandis entered. When you exit from the encrypted filter submode without entering all the mandatory parameters, an error message is printed before exiting the submode. The cipher key, cipher value, and filter hash are the mandatory values. A filter is not configured in this case.

TCDF Packaging Support

TCDFs are FPM filters in XML format. Each TCDF file is designed to filter for a single individual worm or virus. TCDF packaging support provides packages containing at least one or more worm or virus filters and efficiently updates FPM filters as threat characteristics change. When FPM filters are updated, all systems in a network are automatically updated. This behavior reduces the amount of router configuration needed to deploy FRM filters.

To access TCDF packages, configure the router using the **time-range** command to periodically check for package updates. At the specified time, the router connects to the server containing the FPM packages to request the latest version. When the router gets feedback from the server, it compares the FPM package version number from the server with the local FPM package version. If there is an updated package on the server, then the router downloads the package content, replaces the old package with the new package, and updates the local configuration.

Full Packet FPM Search Window Increase

FPM supports searching for patterns up to 256 bytes long anywhere within the entire packet. Also, the number of filters that can be configured per class map is 32. The additional filters can help offset adverse CPU performance that may occur if the “window” for pattern searching is increased. This will also allow FPM users to take advantage of the regular expressions (regex) strings used by Intrusion Prevention Systems (IPS) in their signatures.

Session-based Flexible Packet Matching

FPM works at its best when the filter information exists in all packets of a packet flow. However, if matching contents only exist in a limited number of packets (regex strings and strings in the payload), then FPM can only apply actions to these packets, and miss the other packets in the same packet flow, which are a sequence of packets with the same attributes.

With the introduction of Cisco IOS Release 15.1(3)T, FPM can now match every packet against the filters specified in the class map and pass the match result to consecutive packets of the same network session. If a filter matches with malicious content in the packet’s protocol header or payload, then the required action is taken to resolve the problem.

The **match class session** command configures match criteria that identify a session containing packets of interest, which is then applied to all packets transmitted during the session. The **packet-range** and **byte-range** keywords are used to create a filter mechanism that increases the performance and matching

accuracy of regex-based FPM class maps by classifying traffic that resides in the narrow packet number or byte ranges of each packet flow. If packets go beyond the classification window, then the packet flow can be identified as unknown and packet classification is terminated early to increase performance. For example, a specific application can be blocked efficiently by filtering all packets that belong to this application on a session. These packets are dropped without matching every individual packet with the filters, which improves the performance of a session.

These filters also reduce the number of false positives introduced by general regex-based approaches. For example, internet company messenger traffic can be classified with a string like **intco**, **intcomsg**, and **ic**. These strings are searched for in a packet's payload. These small strings can appear in the packet payload of any other applications, such as e-mail, and can introduce false positives. False positives can be avoided by specifying which regex is searched within which packet of a particular packet flow. See "Creating a Traffic Class for Flexible Packet Matching" for more information.

Once the match criteria are applied to packets belonging to the specific traffic class, these packets can be discarded by configuring the **drop all**command in a policy map. Packets match only on the packet flow entry of an FPM, and skip user-configured classification filters. See "Creating a Traffic Policy for Flexible Packet Matching" for more information.

A match class does not have to be applied exclusively for a regex-based filter. Any FPM filter can be used in the nested match class filter. For example, if the match class **c1** has the filter **match field TCP source-port eq 80**, then the **match class c1 session** command takes the same action for the packets that follow the first matching packet.

How to Configure a Flexible Packet Matching Traffic Class and Traffic Policy

Creating a Traffic Class for Flexible Packet Matching

Note If the PHDF protocol fields are referenced in the access-control classmap, the stack classmap is required in order to make FPM work properly

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **load protocol** *location:filename*
4. **class-map** [**type** {**stack** | **access-control**}] *class-map-name* [**match-all** | **match-any**]
5. **description** *character-string*
6. **match field** *protocol protocol-field* {**eq** [*mask*] | **neq** | [*mask*] | **gt** | **lt** | **range** *range* | **regex** *string*} *value* [**next** *next-protocol*]
7. **match start** {**l2-start** | **l3-start**} **offset** *number* **size** *number* {**eq** | **neq** | **gt** | **lt** | **range** *range* | **regex** *string*} {*value* [*value2*] | [*string*]}
8. **match class** *class-name* [**packet-range** *low high* | **byte-range** *low high*] **session**
9. **exit**
10. **exit**
11. **show class-map** [**type** {**stack** | **access-control**} | *class-map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**load protocol** *location:filename* **Example:** `Router(config)# load protocol disk2:udp.phdf`	(Optional) Loads a PHDF onto a router. • The specified location must be local to the router. **Note** If a PHDF is not loaded, only the **match start** command can be used; that is, you cannot issue the **match field** command. **Note** For the ASR platform, PHDF files should be manually copied (through the **load protocol** command) to the active and standby route processor (RP) file systems.

	Command or Action	Purpose
Step 4	**class-map** [**type** {**stack** | **access-control**}] *class-map-name* [**match-all** | **match-any**] **Example:** `Router(config)# class-map type access-control c1`	Creates a class map to be used for matching packets to a specified class and enters class-map configuration mode. • **type stack** -- Enables FPM to determine the correct protocol stack in which to examine. • **type access-control** -- Determines the exact pattern to look for in the protocol stack of interest. • *class-map-name* -- Can be a maximum of 40 alphanumeric characters. • If **match-all** or **match-any**or are not specified, traffic must match all the match criterion to be classified as part of the traffic class.
Step 5	**description** *character-string* **Example:** `Router(config-cmap)# description “match on slammer packets”`	(Optional) Adds a description to the class map.
Step 6	**match field** *protocol protocol-field* {**eq** [*mask*] | **neq** | [*mask*] | **gt** | **lt** | **range** *range* | **regex** *string*} *value* [**next** *next-protocol*] **Example:** `Router(config-cmap)# match field udp dest-port eq 0x59A`	(Optional) Configures the match criteria for a class map on the basis of the fields defined in the PHDFs. • The **next** *next-protocol* keyword-argument pair is available only after configuring the **class-map type stack** command.
Step 7	**match start** {**l2-start** | **l3-start**} **offset** *number* **size** *number* {**eq** | **neq** | **gt** | **lt** | **range** *range* | **regex** *string*} {*value* [*value2*] | [*string*]} **Example:** `Router(config-cmap)# match start l3-start offset 224 size 4 eq 0x4011010`	(Optional) Configures the match criteria for a class map on the basis of the datagram header (Layer 2) or the network header (Layer 3).

	Command or Action	Purpose
Step 8	**match class** *class-name* [**packet-range** *low high* | **byte-range** *low high*] **session** **Example:** `Router(config-cmap)# match class c2 packet-range 1 5 session`	(Optional) Configures match criteria for a class map that identifies a session (flow) containing packets of interest, which is then applied to all packets transmitted during the session. The **packet-range** and **byte-range** keywords create a filter mechanism that increases the performance and matching accuracy of regex-based FPM class maps by classifying traffic that resides in the narrow packet number or packet byte ranges of each packet flow. When the **session** keyword is used with the *class-name* argument, the classification results are preserved for the subsequent packets of the same packet session. When the **session** keyword is used with the **packet-range** or **byte-range** keywords, the classification results are preserved for the specified packets or bytes of the same packet session.
Step 9	**exit** **Example:** `Router(config-cmap)# exit`	Exits class-map configuration mode.
Step 10	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 11	**show class-map** [**type** {**stack** | **access-control**} | *class-map-name*] **Example:** `Router# show class-map type access-control slammer`	(Optional) Displays configured FPM class maps.

Creating a Traffic Policy for Flexible Packet Matching

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map type access-control** *policy-map-name*
4. **description** *character-string*
5. **class** *class-name* **insert-before** *class-name*
6. **drop** [**all**]
7. **log** [**all**]
8. **service-policy** *policy-map-name*
9. **exit**
10. **interface** *type number*
11. **service-policy type access-control** {**input** | **output**} *policy-map-name*
12. **exit**
13. **exit**
14. **show policy-map** [**type access-control** | **interface** *type number* | **input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map type access-control** *policy-map-name* **Example:** `Router(config)# policy-map type access-control fpm-udp-policy`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy, and enters policy-map configuration mode.

	Command or Action	Purpose
Step 4	**description** *character-string* **Example:** `Router(config-pmap)# description “policy for UDP based attacks”`	(Optional) Adds a description to the policy map.
Step 5	**class** *class-name* **insert-before** *class-name* **Example:** `Router(config-pmap)# class slammer`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command. The **class** command also classifies traffic to the traffic policy and enters policy-map class configuration mode. • The **insert-before** *class-name* keyword and argument adds a class map to any location within the policy map. If this option is not issued, the class map is appended to the end of the policy map.
Step 6	**drop** [**all**] **Example:** `Router(config-pmap-c)# drop all`	(Optional) Configures a traffic class to discard packets belonging to a specific class. The **all** keyword is used to discard the entire stream of packets belonging to the traffic class. If this command is issued, note the following restrictions: • Discarding packets is the only action that can be configured in a traffic class. • When a traffic class is configured with the **drop** command, a “child” (nested) policy cannot be configured for this specific traffic class through the **service policy** command. • Discarding packets cannot be configured for the default class specified via the **class class-default** command. • If the **drop all**command is specified, then this command can only be associated with a **class map type access-control** command.
Step 7	**log** [**all**] **Example:** `Router(config-pmap-c)# log all`	(Optional) Generates log messages for the traffic class. The **all** keyword is used to log the entire stream of discarded packets belonging to the traffic class. This keyword is only available for a class map that is created with the **class-map type access-control** command.
Step 8	**service-policy** *policy-map-name* **Example:** `Router(config-pmap-c)# service policy fpm-udp-policy`	Creates hierarchical service policies.

	Command or Action	Purpose

Step 9	**exit**
	
	Example:
	`Router(config-pmap-c)# exit`
	
	Example:
	`Router(config-pmap)# exit`
		Exits policy-map class configuration mode and policy-map configuration mode.

Step 10	**interface** *type number* **Example:** `Router(config)# interface gigabitEthernet 0/1`	Configures an interface type and enters interface configuration mode.
Step 11	**service-policy type access-control** {**input** | **output**} *policy-map-name* **Example:** `Router(config-if)# service-policy type access-control input fpm-policy`	Specifies the type and the name of the traffic policy to be attached to the input or output direction of an interface.
Step 12	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode.
Step 13	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 14	**show policy-map** [**type access-control** | **interface** *type number* | **input** | **output**] **Example:** `Router# show policy-map type access-control interface gigabitethernet 0/1`	(Optional) Verifies the FPM configuration. **Note** Once a traffic policy is created for FPM, a matched packet can be copied or redirected to a different destination interface.

- Copying a Matched Packet To a Different Destination Interface, page 13
- Redirecting a Matched Packet To a Different Destination Interface, page 15

Copying a Matched Packet To a Different Destination Interface

Perform this task to configure a traffic class to copy packets belonging to a specific class to a different destination interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map type access-control** *policy-map-name*
4. **description** *character-string*
5. **class** *class-name* **insert-before** *class-name*
6. **copy interface** *type number*
7. **service-policy** *policy-map-name*
8. **exit**
9. **interface** *type number*
10. **service-policy type access-control** {**input** | **output**} *policy-map-name*
11. **exit**
12. **exit**
13. **show policy-map type access-control** [**interface** *type number*] [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map type access-control** *policy-map-name* **Example:** `Router(config)# policy-map type access-control fpm-udp-policy`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters policy-map configuration mode.

	Command or Action	Purpose
Step 4	**description** *character-string* **Example:** `Router(config-pmap)# description “policy for UDP based attacks”`	(Optional) Adds a description to the policy map.
Step 5	**class** *class-name* **insert-before** *class-name* **Example:** `Router(config-pmap)# class cmtest`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy. • **insert-before** *class-name* --Adds a class map to any location within the policy map. If this option is not issued, the class map is appended to the end of the policy map.
Step 6	**copy interface** *type number* **Example:** `Router(config-pmap-c)# copy interface FastEthernet 4/15`	(Optional) Configures a traffic class to copy packets belonging to a specific class to a different destination interface. If this command is issued, note the following restrictions: • This command cannot be used with drop or redirect interface command. • This command cannot be configured with a service policy for a stack class. • The packets can only be copied to the following interfaces: Ethernet, Fast Ethernet, Gigabit Ethernet, and Ten Gigabit Ethernet.
Step 7	**service-policy** *policy-map-name* **Example:** `Router(config-pmap-c)# service policy fpm-udp-policy`	Creates hierarchical service policies.

Step 8	**exit**
	
	Example:
	`Router(config-pmap-c)# exit`
	
	Example:
	`Router(config-pmap)# exit`
		Exits policy-map class configuration mode and policy-map configuration mode.

	Command or Action	Purpose
Step 9	**interface** *type number* **Example:** `Router(config)# interface gigabitEthernet 0/1`	Configures an interface type and enters interface configuration mode.
Step 10	**service-policy type access-control** {**input** | **output**} *policy-map-name* **Example:** `Router(config-if)# service-policy type access-control input fpm-policy`	Specifies the type and the name of the traffic policy to be attached to the input or output direction of an interface.
Step 11	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode.
Step 12	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 13	**show policy-map type access-control** [**interface** *type number*] [**input** | **output**] **Example:** `Router# show policy-map type access-control interface gigabit 0/1`	(Optional) Verifies the FPM configuration.

Redirecting a Matched Packet To a Different Destination Interface

Perform this task to configure a traffic class to redirect packets belonging to a specific class to a different destination.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map type access-control** *policy-map-name*
4. **description** *character-string*
5. **class** *class-name* **insert-before** *class-name*
6. **redirect interface** *type number*
7. **service-policy** *policy-map-name*
8. **exit**
9. **interface** *type number*
10. **service-policy type access-control** {**input** | **output**} *policy-map-name*
11. **exit**
12. **exit**
13. **show policy-map type access-control** [**interface** *type number*][**input** | **output**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map type access-control** *policy-map-name* **Example:** `Router(config)# policy-map type access-control fpm-udp policy`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters policy-map configuration mode.
Step 4	**description** *character-string* **Example:** `Router(config-pmap)# description “policy for UDP based attacks”`	(Optional) Adds a description to the policy map.

	Command or Action	Purpose
Step 5	**class** *class-name* **insert-before** *class-name* **Example:** `Router(config-pmap)# class cmtest`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy. • **insert-before** *class-name* --Adds a class map to any location within the policy map. If this option is not issued, the class map is appended to the end of the policy map.
Step 6	**redirect interface** *type number* **Example:** `Router(config-pmap-c)# redirect interface FastEthernet 4/15`	(Optional) Configures a traffic class to redirect packets belonging to a specific class to a different destination interface. If this command is issued, note the following restrictions: • This command cannot be using with the drop or copy interface command. • This command cannot be configured with a service policy for a stack class. • The packets can only be copied to the following interfaces: Ethernet, Fast Ethernet, Gigabit Ethernet, and Ten Gigabit Ethernet.
Step 7	**service-policy** *policy-map-name* **Example:** `Router(config-pmap-c)# service policy fpm-udp-policy`	Creates hierarchical service policies.

Step 8	**exit**
	
	Example:
	`Router(config-pmap-c)# exit`
	
	Example:
	`Router(config-pmap)# exit`
		Exits policy-map class configuration mode and policy-map configuration mode.

Step 9	**interface** *type number* **Example:** `Router(config)# interface gigabitEthernet 0/1`	Configures an interface type and enters interface configuration mode.

	Command or Action	Purpose
Step 10	**service-policy type access-control** {**input** | **output**} *policy-map-name* **Example:** `Router(config-if)# service-policy type access-control input fpm-policy`	Specifies the type and the name of the traffic policy to be attached to the input or output direction of an interface.
Step 11	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode.
Step 12	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 13	**show policy-map type access-control** [**interface** *type number*][**input** | **output**]] **Example:** `Router# show policy-map type access-control interface gigabit 0/1`	(Optional) Verifies the FPM configuration.

Configuring Packaging Support for Flexible Packet Matching

Perform this task to configure FPM packaging support.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **fpm package-info**
4. **time-range** *time-setting*
5. **host** *ip-address*
6. **local-path** *memory-option*
7. **remote-path** *path-name*
8. **exit**
9. **fpm package-group fpm-group-name**
10. **package** *fpm-package-name*
11. **action log**
12. **exit**
13. **auto-load**
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**fpm package-info** **Example:** `Router(config)# fpm package-info`	Enters FPM configuration mode.
Step 4	**time-range** *time-setting* **Example:** `Router(config-fpm-pak-info)# time-range weekly`	Specifies the time interval to check for new FPM packages.

	Command or Action	Purpose
Step 5	**host** *ip-address* **Example:** `Router(config-fpm-pak-info)# host 10.0.0.1`	Specifies the location from where FPM package updates are downloaded.
Step 6	**local-path** *memory-option* **Example:** `Router(config-fpm-pak-info)# local-path flash:`	Specifies where the FPM packages are stored locally.
Step 7	**remote-path** *path-name* **Example:** `Router(config-fpm-pak-info)# remote-path fpm-security`	Specifies the location of the FPM packages on the FPM server.
Step 8	**exit** **Example:** `Router(config-fpm-pak-info)# exit`	Exits FPM configuration.
Step 9	**fpm package-group fpm-group-name** **Example:** `Router(config)# fpm package-group fpm-update`	Specifies an FPM group and enters FPM group definition mode.
Step 10	**package** *fpm-package-name* **Example:** `Router(config-fpm-pak-grp)# package fpm-group-44`	Specifies an FPM package and enters FPM package definition mode.
Step 11	**action log** **Example:** `Router(config-fpm-pak-grp-pak)# action log`	Enables logging for this FPM package.

	Command or Action	Purpose
Step 12	**exit** **Example:** `Router(config-fpm-pak-grp-pak)# exit`	Exits FPM package mode and enters FPM group definition configuration mode.
Step 13	**auto-load** **Example:** `Router(config-fpm-pak-grp)# auto-load`	Enable automatic loading of the FPM package.
Step 14	**end** **Example:** `Router(config-fpm-pak-grp)# end`	Exits FPM configuration mode and enters privileged EXEC mode.

Configuring eTCDF Through the Command-Line Interface

If you have access to an encrypted traffic classification definition file (eTCDF) or if you know valid values to configure encrypted FPM filters, you can configure the same eTCDF through the command-line interface instead of using the preferred method of loading the eTCDF on the router. You can copy the values from the eTCDF by opening the eTCDF in any text editor.

Perform this task to configure eTCDF through the command-line interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map type access-control** [**match-all** | **match-any**] *class-map-name*
4. **match encrypted**
5. **algorithm** *algorithm*
6. **cipherkey** *key-name*
7. **ciphervalue** *contents*
8. **filter-hash** *hash-value*
9. **filter-id** *id-value*
10. **filter-version** *version*
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**class-map type access-control** [**match-all** | **match-any**] *class-map-name* **Example:** `Router(config)# class-map type access-control match-all class1`	Creates a class map to be used for matching packets to a specified class and enters class-map configuration mode.
Step 4	**match encrypted** **Example:** `Router(config-cmap)# match encrypted`	Configures the match criteria for a class map on the basis of encrypted Flexible Packet Matching (FPM) filters and enters the FPM match encryption filter configuration mode.
Step 5	**algorithm** *algorithm* **Example:** `Router(c-map-match-enc-config)# algorithm aes256cbc`	Specifies the algorithm to be used for decrypting the filters.
Step 6	**cipherkey** *key-name* **Example:** `Router(c-map-match-enc-config)# cipherkey realm-abc.sym`	Specifies the symmetric key-name that is used to decrypt the filter.

	Command or Action	Purpose

Step 7	**ciphervalue** *contents*
	
	Example:
	`Router(c-map-match-enc-config)# ciphervalue #2bcXhFL8Ld1v+DqU +dnxgmONCxl4JrYfcLl95xg`
	
	Example:
	`ET0b2B1z0sjoCkozE8YxiH/SXL+eG2wf3ogaA7/Fh`
	
	Example:
	`awIH7OF3tUcS5Jwim/u95Xlzh2RLNw819tuIBCdorV`
	
	Example:
	`Cu0ZzWCF3vqwpGQzaxtSE4sFgPAvSE2LxZc/VT22`
	
	Example:
	`F7EQKBhRo=#`
		Specifies the encrypted filter contents.

Step 8	**filter-hash** *hash-value* **Example:** `Router(c-map-match-enc-config)# filter-hash AABBCCDD11223344`	Specifies the hash for verification and validation of decrypted contents.
Step 9	**filter-id** *id-value* **Example:** `Router(c-map-match-enc-config)# filter-id id2`	Specifies a filter-level ID for encrypted filters.
Step 10	**filter-version** *version* **Example:** `Router(c-map-match-enc-config)# filter-version v1`	Specifies the filter-level version value for the encrypted filter.

	Command or Action	Purpose
Step 11	**end** **Example:** `Router(c-map-match-enc-config)# end`	Exits FPM match encryption filter configuration mode and returns to privileged EXEC mode.

Configuration Examples for Flexible Packet Matching

Example: Configuring FPM for Slammer Packets

The following example shows how to define FPM traffic classes for slammer packets (UDP port 1434). The match criteria defined within the class maps is for slammer packets with an IP length not to exceed 404 bytes, UDP port 1434, and pattern 0x4011010 at 224 bytes from start of IP header. This example also shows how to define the service policy “fpm-policy” and apply it to the Gigabit Ethernet interface. Show commands have been issued to verify the FPM configuration. (Note that PHDFs are not displayed in show output because they are in XML format.)

```
Router(config)# load protocol disk2:ip.phdf
Router(config)# load protocol disk2:udp.phdf
Router(config)# class-map type stack match-all ip-udp
Router(config-cmap)# description “match UDP over IP packets”
Router(config-cmap)# match field ip protocol eq 0x11 next udp
Router(config)# class-map type access-control match-all slammer
Router(config-cmap)# description “match on slammer packets”
Router(config-cmap)# match field udp dest-port eq 0x59A
Router(config-cmap)# match field ip length gt 0x194
Router(config-cmap)# match start l3-start offset 224 size 4 eq 0x4011010
Router(config)# policy-map type access-control fpm-udp-policy
Router(config-pmap)# description “policy for UDP based attacks”
Router(config-pmap)# class slammer
Router(config-pmap-c)# drop
Router(config)# policy-map type access-control fpm-policy
Router(config-pmap)# description “drop worms and malicious attacks”
Router(config-pmap)# class ip-udp
Router(config-pmap-c)# service-policy fpm-udp-policy
Router(config)# interface gigabitEthernet 0/1
Router(config-if)# service-policy type access-control input fpm-policy
Router# show policy-map type access-control interface gigabit 0/1
GigabitEthernet0/1
Service-policy access-control input: fpm-policy
Class-map: ip-udp (match-all)
0 packets, 0 bytes
3 minute offered rate 0 bps
```

```
Match: field IP protocol eq 0x11 next UDP
Service-policy access-control : fpm-udp-policy
Class-map: slammer (match-all)
0 packets, 0 bytes
3 minute offered rate 0 bps, drop rate 0 bps
Match: field UDP dest-port eq 0x59A
Match: field IP length eq 0x194
Match: start l3-start offset 224 size 4 eq 0x4011010
drop
Class-map: class-default (match-any)
0 packets, 0 bytes
3 minute offered rate 0 bps, drop rate 0 bps
Match: any
Class-map: class-default (match-any)
0 packets, 0 bytes
3 minute offered rate 0 bps, drop rate 0 bps
Match: any
Router# show protocol phdf ip
Protocol ID: 1
Protocol name: IP
Description: Definition-for-the-IP-protocol
Original file name: disk2:ip.phdf
Header length: 20
Constraint(s):
Total number of fields: 12
Field id: 0, version, IP-version
Fixed offset. offset 0
Constant length. Length: 4
Field id: 1, ihl, IP-Header-Length
Fixed offset. offset 4
Constant length. Length: 4
Field id: 2, tos, IP-Type-of-Service
Fixed offset. offset 8
Constant length. Length: 8
Field id: 3, length, IP-Total-Length
Fixed offset. offset 16
Constant length. Length: 16
Field id: 4, identification, IP-Identification
Fixed offset. offset 32
Constant length. Length: 16
Field id: 5, flags, IP-Fragmentation-Flags
Fixed offset. offset 48
Constant length. Length: 3
Field id: 6, fragment-offset, IP-Fragmentation-Offset
Fixed offset. offset 51
Constant length. Length: l3
Field id: 7, ttl, Definition-for-the-IP-TTL
Fixed offset. offset 64
Constant length. Length: 8
Field id: 8, protocol, IP-Protocol
Fixed offset. offset 72
Constant length. Length: 8
Field id: 9, checksum, IP-Header-Checksum
Fixed offset. offset 80
Constant length. Length: 16
Field id: 10, source-addr, IP-Source-Address
Fixed offset. offset 96
Constant length. Length: 32
Field id: 11, dest-addr, IP-Destination-Address
Fixed offset. offset 128
Constant length. Length: 32
Router# show protocol phdf udp
Protocol ID: 3
Protocol name: UDP
Description: UDP-Protocol
Original file name: disk2:udp.phdf
Header length: 8
Constraint(s):
Total number of fields: 4
Field id: 0, source-port, UDP-Source-Port
Fixed offset. offset 0
Constant length. Length: 16
Field id: 1, dest-port, UDP-Destination-Port
```

```
Fixed offset. offset 16
Constant length. Length: 16
Field id: 2, length, UDP-Length
Fixed offset. offset 32
Constant length. Length: 16
Field id: 3, checksum, UDP-Checksum
Fixed offset. offset 48
Constant length. Length: 16
```

Example: Configuring FPM for Blaster Packets

The following example shows how to configure FPM for blaster packets. The class map contains the following match criteria: TCP port 135, 4444 or UDP port 69; and pattern 0x0030 at 3 bytes from the start of the IP header.

```
Router(config)# load protocol disk2:ip.phdf
Router(config)# load protocol disk2:tcp.phdf
Router(config)# load protocol disk2:udp.phdf

Router(config)# class-map type stack match-all ip-tcp
Router(config-cmap)# match field ip protocol eq 0x6 next tcp
Router(config)# class-map type stack match-all ip-udp
Router(config-cmap)# match field ip protocol eq 0x11 next udp
Router(config)# class-map type access-control match-all blaster1
Router(config-cmap)# match field tcp dest-port eq 135
Router(config-cmap)# match start l3-start offset 3 size 2 eq 0x0030
Router(config)# class-map type access-control match-all blaster2
Router(config-cmap)# match field tcp dest-port eq 4444
Router(config-cmap)# match start l3-start offset 3 size 2 eq 0x0030
Router(config)# class-map type access-control match-all blaster3
Router(config-cmap)# match field udp dest-port eq 69
Router(config-cmap)# match start l3-start offset 3 size 2 eq 0x0030
Router(config)# policy-map type access-control fpm-tcp-policy
Router(config-pmap)# class blaster1
Router(config-pmap-c)# drop
Router(config-pmap-c)# class blaster2
Router(config-pmap-c)# drop
Router(config)# policy-map type access-control fpm-udp-policy
Router(config-pmap)# class blaster3
Router(config-pmap-c)# drop
Router(config)# policy-map type access-control fpm-policy
Router(config-pmap)# class ip-tcp
Router(config-pmap-c)# service-policy fpm-tcp-policy
Router(config-pmap)# class ip-udp
Router(config-pmap-c)# service-policy fpm-udp-policy
Router(config)# interface gigabitEthernet 0/1
Router(config-if)# service-policy type access-control input fpm-policy
```

Example: Configuring FPM for MyDoom Packets

The following example shows how to configure FPM for MyDoom packets. The match criteria is as follows:

- 90 > IP length > 44
- pattern 0x47455420 at 40 bytes from start of IP header

or

- IP length > 44
- pattern 0x6d3a3830 at 48 bytes from start of IP header
- pattern 0x47455420 at 40 bytes from start of IP header

```
Router(config)# load protocol disk2:ip.phdf
Router(config)# load protocol disk2:tcp.phdf
```

```
Router(config)# class-map type stack match-all ip-tcp
Router(config-cmap)# match field ip protocol eq 0x6 next tcp
Router(config)# class-map type access-control match-all mydoom1
Router(config-cmap)# match field ip length gt 44
Router(config-cmap)# match field ip length lt 90
Router(config-cmap)# match start l3-start offset 40 size 4 eq 0x47455420
Router(config)# class-map type access-control match-all mydoom2
Router(config-cmap)# match field ip length gt 44
Router(config-cmap)# match start l3-start offset 40 size 4 eq 0x47455420
Router(config-cmap)# match start l3-start offset 48 size 4 eq 0x6d3a3830
Router(config)# policy-map type access-control fpm-tcp-policy
Router(config-pmap)# class mydoom1
Router(config-pmap-c)# drop
Router(config-pmap-c)# class mydoom2
Router(config-pmap-c)# drop
Router(config)# policy-map type access-control fpm-policy
Router(config-pmap)# class ip-tcp
Router(config-pmap-c)# service-policy fpm-tcp-policy
Router(config)# interface gigabitEthernet 0/1
Router(config-if)# service-policy type access-control input fpm-policy
```

Example: Configuring and Verifying FPM on ASR Platform

The following example shows how to configure FPM on the ASR platform.

```
load protocol bootflash:ip.phdf
load protocol bootflash:tcp.phdf
class-map type stack match-all ip-tcp
 match field IP protocol eq 6 next TCP
class-map type access-control match-all test-class
 match field TCP dest-port gt 10
 match start l3-start offset 40 size 32 regex "ABCD"
policy-map type access-control child
 class test-class
  drop
policy-map type access-control parent
 class ip-tcp
  service-policy child
interface GigabitEthernet0/3/0
 ip address 10.1.1.1 255.0.0.0
 service-policy type access-control input parent
```

In the following sample output, all TCP packets are seen under the class map named ip_tcp and all packets matching the specific pattern are seen under the class map named test_class. TCP packets without the specific pattern are seen under the child policy named class-default, while all non-TCP packets are seen under the parent policy named class-default. (The counter is 0 in this example.)

```
Router# show policy-map type access-control interface gig0/3/0
GigabitEthernet0/3/0
 Service-policy access-control input: parent
  Class-map: ip_tcp (match-all)
  2024995578 packets, 170099628552 bytes
  5 minute offered rate 775915000 bps
  Match: field IP version eq 4
  Match: field IP ihl eq 5
  Match: field IP protocol eq 6 next TCP
 Service-policy access-control : child
 Class-map: test_class (match-all)
  1598134279 packets, 134243279436 bytes
  5 minute offered rate 771012000 bps, drop rate 771012000 bps
  Match: field TCP dest-port gt 10
  Match: start l3-start offset 40 size 32 regex "ABCD"
 drop
 Class-map: class-default (match-any)
  426861294 packets, 35856348696 bytes
  5 minute offered rate 4846000 bps, drop rate 0 bps
  Match: any
 Class-map: class-default (match-any)
  0 packets, 0 bytes
```

```
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any
Router#
```

Example: Configuring Session-based FPM

The following example shows how to configure a class map and policy map to specify the protocol stack class, the match criteria and action to take, and a combination of classes using session-based (flow-based) and nonsession-based actions. The **drop all** command is associated with the action to be taken on the policy.

```
Router(config)# class-map type access-control match-all my-HTTP
Router(config-cm)# match field tcp destport eq 8080
Router(config-cm)# match start tcp payload-start offset 20 size 10 regex “GET”
Router(config)# class-map type access-control match-all my-FTP
Router(config-cmap)# match field tcp destport eq 21
Router(config)# class-map type access-control match all class1
Router(config-cmap)# match class my-HTTP session
Router(config-cmap)# match start tcp payload-start offset 40 size 20 regex “abc.*def”
Router(config)# policy-map type access-control my_http_policy
Router(config-pmap)# class class1
Router(config-pmap-c)# drop all
Router(config)# interface gigabitEthernet 0/1
Router(config-if)# service-policy type access-control input my_http_policy
```

Example: Configuring Session-based FPM with a Filter for Increased Performance and Accuracy

The following example shows how to configure a class map and policy map to specify the protocol stack class, the match criteria and action to take, and a combination of classes using session-based (flow-based) and nonsession-based actions. However, this example uses the **match class packet-range** command, which acts as a filter mechanism to increases the performance and matching accuracy of the regex-based FPM class map.

```
Router(config)# load disk2:ip.phdf
Router(config)# load protocol disk2:tcp.phdf
Router(config)# class-map type stack match-all ip_tcp
Router(config-cmap)# description “match TCP over IP packets”
Router(config-cmap)# match field ip protocol eq 6 next tcp
Router(config)# class-map type access-control match-all WM
Router(config-cmap) # match start tcp payload-start offset 20 size 20 regex
“.*(WEBCO|WMSG|WPNS).......[LWT].*\xc0\x80”
Router(config)# class-map type access-control match-all wtube
Router(config-cmap) # match start tcp payload-start offset 20 size 20 regex
“.*GET\x20.*HTTP\x2f(0\.9|1\.0|1\.1)\x0d\x0aHost:\x20webtube.com\x0d\x0a”
Router(config)# class-map type access-control match-all doom
Router(config-cmap) # match start tcp payload-start offset 20 size 20 string virus
Router(config)# class-map type access-control match-all class_webco
Router(config-cmap)# match class WM session
Router(config-cmap)# match field ip length eq 0x194
Router(config-cmap)# match start network-start offset 224 size 4 eq 0x4011010
Router(config)# class-map type access-control match-all class_webtube
Router(config-cmap)# match class wtube packet-range 1 5 session
Router(config-cmap)# match class doom session
Router(config-cmap)# match field ip length eq 0x194
Router(config-cmap)# match start network-start offset 224 size 4 eq 0x4011010
Router(config)# policy-map type access-control my_policy
Router(config-pmap)# class class_webco
Router(config-pmap-c)# log
Router(config)# policy-map type access-control my_policy
Router(config-pmap)# class class_webtube
Router(config-pmap-c)# drop all
Router(config)# policy-map type access-control P1
Router(config-pmap)# class ip_tcp
```

```
Router(config-pmap-c)# service-policy my_policy
Router(config)# interface gigabitEthernet 0/1
Router(config-if)# service-policy type access-control input P1
```

Example: Verifying FPM Package Support

The following example shows how to verify FPM Package support.

```
Router# show fpm package-info
 fpm package-info
  host 10.0.0.1
  remote-path fpm-group/
  local-path archive/
  user cisco
  password
  protocol
  time-range weekly
Router# show fpm package-group
 group name: fpm-weekly-update
  auto-load
  fpm package: fpm-package-45
  fpm package: fpm-group-secure
   package action: log
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Security commands	*Cisco IOS Security Command Reference*
Configuring FPM using traffic classification definition files.	"Flexible Packet Matching XML Configuration" module in the *Cisco IOS Security Configuration Guide: Securing the Data Plane*
Complete suite of quality of service (QoS) commands	*Cisco IOS Quality of Service Solutions Command Reference*

Standards

Standards	Title
None	--

MIBs

MIBs	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Flexible Packet Matching

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1 Feature Information for Flexible Packet Matching

Feature Name	Releases	Feature Information
Flexible Packet Matching	12.4(4)T 12.2(18)ZY	FPM is a packet classification feature that allows users to define one or more classes of network traffic by pairing a set of standard matching operators with user-defined protocol header fields. In Cisco IOS Release 12.2(18)ZY, FPM was implemented on the Catalyst 6500 series of switches equipped with the PISA. The following commands were introduced or modified: **class** , **class-map**, **copy interface**, **debug fpm event**, **description**, **load protocol**, **match field**, **match start**, **policy-map**, **service-policy**, **show class-map**, **show policy-map interface**, **redirect interface**, **show protocol phdf**.
FPM Full Packet Filtering	12.4(15)T	In Cisco IOS Release 12.4(15)T, FPM supports searching for patterns up to 56 bytes long anywhere within the entire packet. Prior to 12.4(15)T, FPM only supported searching for patterns up to 32 bytes long within the first 256 bytes of the packet.

Feature Name	Releases	Feature Information
Enhance FPM Search Window Size to 128 Bytes	12.2(18)ZYA	FPM supports searching for patterns up to 128 bytes long anywhere within the entire packet. Also, the number of filters that can be configured per class map has increased from 8 to 32. The additional filters can help offset adverse CPU performance that may occur if the "window" for pattern searching is increased. (However, some optimizations for better performance are possible with **match-any** type of class maps that have filters starting at same the same offset and the same size.)
FPM Copy or Redirect Matched Packets	12.2(18)ZYA1	When a match of the policy is found, the packet can be redirected to a different destination or a copy of the packet can be sent to a different destination. This is possible with the **copy interface** and **redirect interface** commands introduced in this release. The actions supported in this release are copy, copy and log, drop, drop and log, log, redirect, redirect and log.
FPM--Packaging, eTCDF, and Full Packet Search Enhancements	15.0(1)M	FPM--Packaging, eTCDF and Full Packet Search Enhancements provide preconfigured FPM filters written in XML format which can be directly loaded onto a router. The following commands were introduced or modified: **algorithm**, **cipherkey**, **ciphervalue**, **filter-hash**, **filter-id**, **filter-version**, **fpm package-group**, **fpm package-info**, **show fpm-package-group**, **match encrypted**, **show fpm package-info**.

Feature Name	Releases	Feature Information
Session-based Flexible Packet Matching	15.1(3)T	With the introduction of Cisco IOS Release 15.1(3)T, FPM can now match every packet against the filters specified in the class map and passes the match result to consecutive packets of the same network session. If a filter matches with malicious content in the packet's protocol header or payload, then the required action is taken to resolve the problem. The following commands were introduced or modified: **match class session**, **drop**, **log**.