Flexible Packet Matching XML Configuration

Last Updated: January 19, 2012

The Flexible Packet Matching XML Configuration feature allows the use of eXtensible Markup Language (XML) to define traffic classes and actions (policies) to assist in blocking network attacks. The XML file used by Flexible Packet Matching (FPM) is called the traffic classification definition file (TCDF). The TCDF gives you an alternative to the command-line interface (CLI) as a method to define traffic classification behavior. Traffic classification behavior is identical regardless of the method you use.

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.

Prerequisites for the Flexible Packet Matching XML Configuration

The Flexible Packet Matching XML Configuration feature has the following prerequisites:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- A protocol header definition file (PHDF) relevant to the TCDF must be loaded on the router.
- Although access to an XML editor is not required, using one might make the creation of the TCDF easier.
- You must be familiar with XML file syntax.

Restrictions for the Flexible Packet Matching XML Configuration

The Flexible Packet Matching XML Configuration has the following restrictions:

- The FPM TCDF 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, when using the FPM TCDF, port numbers must be explicitly specified.
- FPM cannot perform IP fragmentation or TCP flow reassembly.

Information About the Flexible Packet Matching XML Configuration

Before you create and load the TCDF XML configuration files for use with FPM, you should understand the following concepts.

- Traffic Classification Definition Files for the Flexible Packet Matching XML Configuration, page 2
- Protocol Header Definition Files for Traffic Classification Definitions, page 3
- Traffic Classification Description File Format and Use , page 3
- Traffic Class Definitions for a Traffic Classification Definition File , page 4
- Policy Definitions for a Traffic Classification Definition File, page 6

Traffic Classification Definition Files for the Flexible Packet Matching XML Configuration

FPM uses a TCDF to define policies that can block attacks on the network. FPM is a packet classification feature that allows users to define one or more classes of network traffic by pairing a rich set of standard matching operators with user-defined protocol header fields. FPM users can create their own stateless packet classification criteria and define policies with multiple actions (such as drop, log, or send Internet Control Message Protocol [ICMP] unreachable) to immediately block new viruses, worms, and attacks on the network.

Before the release of the Flexible Packet Matching XML Configuration feature, FPM defined traffic classes (class maps), policies (policy maps), and service policies (attach policy maps to a class maps) through the use of CLI commands. 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.

For more information on FPM, see the "Flexible Packet Matching" feature module.

Protocol Header Definition Files for Traffic Classification Definitions

TCDFs require that a relevant PHDF is already loaded on the device. A PHDF defines each field contained in the header of a particular protocol. Each field is described with a name, optional comment, an offset (the location of the protocol header field in relation to the start of the protocol header), and the length of the field. The total length is specified at the end of each PHDF.

The description of a traffic class in a TCDF file can contain header fields defined in a PHDF. If the PHDF is loaded on the router, the class specification to match begins with a list of the protocol headers in the packet. In the TCDF, the traffic class is associated with a policy that binds the match to an action, such as drop, log, or send ICMP unreachable.

FPM provides ready-made definitions for these standard protocols, which can be loaded onto the router with the **load protocol** command: ether.phdf, ip.phdf, tcp.phdf, and udp.phdf. You can also write your own custom PHDFs using XML if one is required for the TCDF.

Note Because PHDFs are defined via XML, they are not shown in a running configuration.

For more information about PHDFs, see the "Flexible Packet Matching" feature module .

Traffic Classification Description File Format and Use

In the TCDF, you can define one or more classes of traffic and policies that describe specified actions for each class of traffic. The TCDF is an XML file that you create in a text file or with an XML editor. The file that you create must have a filename that has the .tcdf extension.

The TCDF has the following basic format. XML tags are shown in bold text for example purposes only.

```
<tdcf
>
       <class
 ...> ... </class
>
          ...
       <policy
> ... </policy
>
          ...
</tdcf
>
```

For a traffic class, you can identify a match for any field or fields against any part of the packet.

Note FPM is stateless and cannot be used to mitigate an attack that requires stateful classification, that is classify across IP fragments, across packets in a TCP stream, or peer-to-peer protocol elements.

Policies can be anything from access control, quality of service (QoS), or even routing decisions. For FPM, the associated actions (policies) might include permit, drop, log, or send ICMP unreachable.

Once loaded, the TCDF-defined classes and policies can be applied to any interface or subinterface and behave in an identical manner as the CLI-defined classes and policies. You can define policies in the TCDF and apply then to any entry point to the network to block new attacks.

Traffic Class Definitions for a Traffic Classification Definition File

A class can be any traffic stream of interest. You define a traffic stream of interest by matching a particular interface or port, a source address or destination IP address, a protocol or an application. The following sections contain information you should understand before you define the traffic class in the TCDF for FPM configuration:

Class Element Attributes for a Traffic Classification Definition File

The table below lists and describes the attributes that you can associate with the **class** element in a TCDF for the FPM XML configuration. The **class** element contains attributes you can use to specify the traffic class name, its description and type, where to look in the packet, what kind of match, and when the actions should apply to the traffic.

Table 1 Attributes for Use with the Class Element in a TCDF for the FPM XMLConfiguration

Attribute Name	Use	Type
name (required)	Specifies the name of the class. **Note** When you use the class element inside policy elements, you need specify the name attribute only.	String
type (required)	Specifies the type of class.	Keywords: stack or access-control
stack start	Specifies where to look in the packet. By default, the match starts at Layer 3.	Keyword: l2-start
match	Specifies the type of match to be performed on the class.	Keywords: all or any • all--All class matches must be met to perform the policy actions. • any--One or more matches within the class must be met to perform the policy actions.
undo	Directs the device to remove the class-map when set to true.	Keywords: true or false

For example, XML syntax for a stack class describing an IP, User Datagram Protocol (UDP), Simple Management Protocol (SNMP) stack might look like this:

```
<class
```

```
 name
="snmp-stack" type
="stack">
    <match
>
       <eq
 field
="ip.protocol" value="x"></eq
>
       <eq
 field
="udp.dport" value
="161"></eq
>
    </match
>
</class
>
```

Match Element for a Traffic Classification Definition File

The **match** element in the TCDF for FPM XML configuration contains **operator** elements. **Operator** elements are the following: **eq** (equal to), **neq** (not equal to), **lt** (less than), **gt** (greater than), **range** (a value in a specific range, for example, **range** 1 - 25), and **regex** (regular expression string with a maximum length of 32 characters).

In following sections, these various operators are collectively called the operator element.

Operator Element Attributes for a Traffic Classification Definition File

The table below lists and describes direct matching attributes that you can associate with the **operator**element in a TCDF for the FPM XML configuration.

Table 2 Direct Matching Attributes to Use with a Match Element in a TCDF for the FPM XML Configuration

Attribute Name	Use	Type
start	Begin the match on a predefined keyword or **Protocol.Field**, if given.	Keyword: l2-start or l3-start Otherwise, a field of a protocol as defined in the PHDF, for example, the source field in the IP protocol.
offset	Used with start attribute. Offset from the start point.	Hexadecimal or decimal number, or string constants, **Protocol.Field**, or combination of a constant and **Protocol.Field** with +, -, *, /, &, or |.
size	Used together with start and offset attributes. How much to match.	Specifies the size of the match in bytes.

Attribute Name	Use	Type
mask	Number specifying bits to be matched in protocol or field attributes. Used exclusively with field type of bitset to specify the bits of interest in a bit map.	Decimal or hexadecimal number
value	Value on which to match.	String, number, or regular expression
field	Specifies the name of the field to be compared.	Name of field as defined in the PHDF
next	Identifies the next layer of the protocol. This attribute can be used only in stack type classes.	Keyword that is the name of a protocol defined in the PHDF.
undo	Directs the device to remove the particular match operator when set to true.	Keywords: true or false

Policy Definitions for a Traffic Classification Definition File

A policy is any action that you apply to a class. You should understand the following information before defining the policy in a TCDF for the FPM XML configuration:

- Policy Element Attributes for a Traffic Classification Definition File, page 6
- Action Element for a Traffic Classification Definition File, page 7

Policy Element Attributes for a Traffic Classification Definition File

Policies can be anything from access control, QoS, or even routing decisions. For FPM, the associated actions or policies might include drop, log, or send ICMP unreachable. Policies describe the action to take to mitigate attacks on the network.

The table below lists and describes the attributes that you can use with the **policy** element in the TDCF for FPM XML configuration.

Table 3 Attributes for Use with the Policy Element in a TCDF for the FPM XML Configuration

Attribute Name	Use	Type
name	Name of the policy.	String
type	Specifies the type of policy map.	Keyword: access-control
undo	Directs the device to remove the policy map when set to true.	Keywords: true or false

The policy name in this example is sql-slammer, and the action defined for the policy is to drop the packet. This action is to be applied to the class that has the same name as the policy (class name= “sql-slammer”).

```
<policy
 name
="sql-slammer“>
    <class
 name
="sql-slammer”></class
>
    <action
>drop</action
>
</policy
>
```

Action Element for a Traffic Classification Definition File

The **action** element is used to specify actions to associate with a policy. The policy with the **action** element is applied to a defined class. The **action** element can contain any of the following: permit, drop, Log, SendBackIcmp, set, RateLimit, alarm, ResetTcpConnection, and DropFlow. For example:

```
<action
>
   log
</action
>
```

How to Create and Load Traffic Classification Definition Files for the FPM XML Configuration

Perform the following tasks to create and load TCDFs for the FPM XML configuration. You can define traffic classes and policies with multiple actions (such as drop, log, or send Internet Control Message Protocol [ICMP] unreachable) in a TCDF to assist in the blocking of new viruses, worms, and attacks on the network.

- Creating a Traffic Classification Definition File for the FPM XML Configuration, page 7
- Loading a Traffic Classification Definition File for the FPM XML Configuration , page 10
- Associating a Traffic Classification Definition File with an Interface or Subinterface, page 12
- Displaying TCDF-Defined Traffic Classes and Policies, page 13

Creating a Traffic Classification Definition File for the FPM XML Configuration

Perform the following task to create a TCDF for FPM XML configuration. The TCDF is used to define traffic classes and the associated policies with specified actions for the purpose of blocking new viruses, worms, and attacks on the network.

The TCDF is configured in a text or XML editor. The syntax of the TCDF must comply with the XML Version 1.0 syntax and the TCDF schema. For information about Version 1.0 XML syntax, see the document at the following url:

http://www.w3.org/TR/REC-xml/

- Traffic Classification Definition File Syntax Guidelines , page 8

Traffic Classification Definition File Syntax Guidelines

The following list describes required and optional syntax for the TCDF:

- The TCDF filename must end in the .tcdf extension, for example, sql_slammer.tcdf.
- The TCDF contains descriptions for one or more traffic classes and one or more policy actions.
- The file is encoded in the XML notation.
- The TCDF file should begin with the following version encoding:

<**?xml version="1.0" encoding="UTF-8"?**>

SUMMARY STEPS

1. Open a text file or an XML editor and begin the file with the XML version and encoding declaration.
2. Identify the file as a TCDF. For example:
3. Define the traffic class of interest.
4. Identify matching criteria for the defined classes of traffic. For example:
5. Define the action to apply to the defined class. For example:
6. End the traffic classification definition. For example:
7. Save the TCDF file with a filename that has a .tcdf extension, for example: slammer.tcdf.

DETAILED STEPS

Step 1 Open a text file or an XML editor and begin the file with the XML version and encoding declaration.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Step 2 Identify the file as a TCDF. For example:

Example:

```
<tcdf
>
```

Step 3 Define the traffic class of interest.
For example, a stack class describing an IP and UDP stack might be described as follows. In this example, the name of the traffic class is “ip-udp,” and the class type is “stack.”

Example:

```
<class

name
=”ip-udp”
type
=”stack”></class
>
```

In the following example, the name of the traffic class is slammer, the class type is access control, and the match criteria is all:

Example:

```
<class
 name="slammer
" type
="access-control" match
="all"></class
>
```

Step 4 Identify matching criteria for the defined classes of traffic. For example:

Example:

```
    <class

name
="ip-udp"
type
="stack">
       <match
>
          <eq

field
="ip.protocol"
value
="0x11"
next
="udp"></eq
>
       </match
>
    </c
lass
>
    <class
 name="slammer
" type
="access-control" match
="all">
       <match
>
          <eq
 field
="udp.dest-port" value
="0x59A"></eq
>
          <eq
 field
="ip.length" value
="0x194"></eq
>
          <eq
 start
="l3-start" offset
="224" size
="4" value
="0x00401010"></eq
>
       </match
>
    </class
>
```

The traffic of interest in this TCDF matches fields defined in the PHDF files, ip.phdf and udp.phdf. The matching criteria for slammer packets is a UDP destination port number 1434 (0x59A), an IP length not to exceed 404 (0x194) bytes, and a Layer 3 position with a pattern 0x00401010 at 224 bytes from start (offset) of the IP header.

Step 5 Define the action to apply to the defined class. For example:

Example:

```
<policy
 name
="fpm-udp-policy">
    <class
 name
="slammer"></class
>
    <action
>Drop</action
>
</policy
>
```

The policy name in this example is fpm-udp-policy, and the action defined for the policy is to drop the packet. This action is to be applied to the class that has the name slammer.

Step 6 End the traffic classification definition. For example:

Example:

```
</tcdf
>
```

Step 7 Save the TCDF file with a filename that has a .tcdf extension, for example: slammer.tcdf.

Loading a Traffic Classification Definition File for the FPM XML Configuration

Perform this task to load a TCDF for the FPM XML configuration. After the TCDF is successfully loaded, you can use service-policy CLI to attach TCDF policies to a specific interface or interfaces (see "Associating a Traffic Classification Definition File with an Interface or Subinterface").

SUMMARY STEPS

1. **enable**
2. **show protocol phdf** *protocol-name*
3. **configure terminal**
4. **load protocol** *location:filename*
5. **load classification** *location* **:** *filename*
6. **end**
7. **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	**show protocol phdf** *protocol-name* **Example:** `Router# show protocol phdf ip`	Displays protocol information from a specific PHDF. • Use this command to verify that a PHDF file relevant to the TCDF is loaded on the device.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**load protocol** *location:filename* **Example:** `Router(config)# load protocol localdisk1:ip.phdf`	(Optional) Loads a PHDF onto a router. • The specified location must be local to the router. **Note** If the required PHDF is already loaded on the router (see Step 2), skip this step and proceed to Step 5).
Step 5	**load classification** *location* **:** *filename* **Example:** `Router(config)# load classification localdisk1:slammer.tcdf`	Loads a TCDF onto a router. • The specified location must be local to the router.
Step 6	**end** **Example:** `Router(config)# end`	Exits to privileged EXEC mode.
Step 7	**show class-map** [type {**stack** | **access-control**}] [*class-map-name*] **Example:** `Router# show class-map sql-slammer`	(Optional) Displays a class map and its matching criteria. • Use this command to verify that a class defined in the TCDF file is available on the device. • The *class-map-name* argument is the name of a class in the TCDF.

Examples

The following is sample output from a **show class-map** command that displays the traffic classes defined in the TCDF after it is loaded on the router:

```
Router# show class-map
.
.
.
class-map type stack match-all ip-udp
   match field IP protocol eq 0x11 next UDP
class-map type access-control match-all slammer
   match field UDP dest-port eq 0x59A
   match field IP length eq 0x194
   match start l3-start offset 224 size 4 eq 0x4011010
.
.
.
```

- What to Do Next, page 12

What to Do Next

After you have defined the TCDF, you must apply that policy to an interface as shown in the following task "Associating a Traffic Classification Definition File with an Interface or Subinterface."

Associating a Traffic Classification Definition File with an Interface or Subinterface

Perform the following task to associate a TCDF with an interface or subinterface.

After the TCDF is loaded, traffic classification behavior defined using the TCDF is identical to the same behavior defined using the CLI.

The TCDP and FPM must be configured on the device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot / port*
4. **service-policy type access-control**] {**input** | **output**} *policy-map-name*
5. **end**
6. **show policy-map interface type access-control**] *interface-name slot/port*[**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot / port* **Example:** `Router(config)# interface gigabitEthernet 0/1`	Configures an interface type and enters interface configuration mode.
Step 4	**service-policy type access-control**] {**input** | **output**} *policy-map-name* **Example:** `Router(config-if)# service-policy type access-control input sql-slammer`	Specifies the type and the name of the traffic policy to be attached to the input or output direction of an interface. • The *policy-map-name* argument is the name of a policy in the TCDF.
Step 5	**end** **Example:** `Router(config-if)# end`	Exits to privileged EXEC mode.
Step 6	**show policy-map interface type access-control**] *interface-name slot/port*[**input** | **output**] **Example:** `Router# show policy-map interface gigabitEthernet 0/1`	(Optional) Displays the packet statistics of all classes that are configured for all service policies either on the specified interface or subinterface. • Use this command to verify that policy defined in TCDF is associated with the named interface.

Displaying TCDF-Defined Traffic Classes and Policies

Perform this task to display TCDF-defined traffic classes and policies.

SUMMARY STEPS

1. **enable**
2. **show class-map [type { stack** | **access-control]** [*class-map-name*
3. **show class-map type stack** [*class-map name*
4. **show class-map type access-control** *class-map-name*
5. **show policy-map** [*policy-map*]
6. **exit**

DETAILED STEPS

Step 1 **enable**
Use this command to enable privileged EXEC mode. Enter your password if prompted. For example:

Example:

```
Router> enable
Router#
```

Step 2 **show class-map [type { stack** | **access-control]** [*class-map-name*
Use this command to verify that a class defined in the TCDF file is available on the device. For example:

Example:

```
Router# show class-map
.
.
.
class-map type stack match-all ip-udp
   match field IP protocol eq 0x11 next UDP
class-map type access-control match-all slammer
   match field UDP dest-port eq 0x59A
   match field IP length eq 0x194
   match start l3-start offset 224 size 4 eq 0x4011010
.
.
.
```

Step 3 **show class-map type stack** [*class-map name*
Use this command to display the stack type defined for the class of traffic in the TCDF file. For example:

Example:

```
Router# show class-map type stack ip-udp
class-map type stack match-all ip-udp
   match field IP protocol eq 0x11 next UDP
```

Step 4 **show class-map type access-control** *class-map-name*
Use this command to display the access type defined for the class in the TCDF file. For example:

Example:

```
Router# show class-map type access-control slammer
class-map type access-control match-all slammer
   match field UDP dest-port eq 0x59A
   match field IP length eq 0x194
   match start l3-start offset 224 size 4 eq 0x4011010
```

Step 5 **show policy-map** [*policy-map*]
Use this command to display the contents of a policy map defined in the TCDF. For example:

Example:

```
Router# show policy-map fpm-udp-policy
policy-map type access-control fpm-udp-policy
```

```
  class slammer
    drop
```

Step 6 **exit**

Use this command to exit to user EXEC mode. For example:

Example:

```
Router# exit
Router>
```

Configuration Examples for Creating and Loading Traffic Classification Definition Files

Note The TCDF files are created in a text file or with an XML editor. In the following examples, XML tags are shown in bold text and field names in italic text. The values for the attributes are entered in quotation marks ("value").

- Example: Configuring FPM for Slammer Packets, page 15
- Example: Configuring FPM for MyDoom Packets , page 17

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 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
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Related Topic	Document Title
Security commands	*Cisco IOS Security Command Reference*
Additional configuration information for class maps and policy maps	"Applying QoS Features Using the MQC"
Information about and configuration tasks for FPM	"Flexible Packet Matching"

MIBs

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

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 XML Configuration

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 4 Feature Information for Flexible Packet Matching XML Configuration

Feature Name	Releases	Feature Information
Flexible Packet Matching XML Configuration	12.4(6)T	The Flexible Packet Matching XML Configuration feature provides an Extensible Markup Language (XML)-based configuration file for Flexible Packet Matching (FPM) that can be used to define traffic classes and actions (policies) to assist in the blocking of attacks on a network. The XML file used by FPM is called the traffic classification definition file (TCDF). The TCDF gives you an alternative to the command-line interface (CLI) as a method to define traffic classification behavior. Traffic classification behavior is identical regardless of the method you use. The following command was introduced by this feature: **load classification**.

Glossary

FPM --Flexible Packet Matching. Packet classification feature that allows users to define one or more classes of network traffic by pairing a rich set of standard matching operators with user-defined protocol header fields.

packet --Logical grouping of information that includes a header containing control information and (usually) user data. Packets most often are used to refer to network layer units of data. The terms datagram, frame, message, and segment also are used to describe logical information groupings at various layers of the OSI reference model and in various technology circles.

stateful classification --Classification that requires state maintenance to identify classes of packets, for example, classifying across IP fragments, classifying across packets in a TCP stream, or classifying peer-to-peer protocols.

stateless classification --Classification that supports a match on any field or fields anywhere in Layer 2 to Layer 7 within the packet. Stateless classification can identify a packet as belonging to a class while utilizing no information other than what is in the packet itself and the class specification.

TCDF --traffic classification definition file. Extensible Markup Language (XML) file created for the purpose of defining traffic classes and policies for Flexible Packet Matching (FPM) that can assist in the blocking of attacks on the network.

XML --eXtensible Markup Language. Standard maintained by the World Wide Web Consortium (W3C). It defines a syntax that lets you create markup languages to specify information structures, which define the

type of information, for example, subscriber name or address, not how the information looks (bold, italic, and so on). External processes can manipulate these information structures and publish them in a variety of formats. Text markup language designed to enable the use of SGML on the World Wide Web. XML allows you to define your own customized markup language.