White Paper

Flexible Packet Matching Deployment Guide

Networks are experiencing increasing sophisticated attacks that require mitigating tools that are as flexible as possible.

OVERVIEW

Flexible Packet Matching (FPM) provides a means to implement network-based blocking of known attack vectors for the advanced user. FPM is the next generation access control list (ACL) pattern matching tool for more thorough and customized packet filters. The technology provides the ability to match on arbitrary bits of a packet at arbitrary depth in the packet header and payload. It removes the constraints to specific fields that previously had limited packet inspection. FPM is provisioned using CLI and off-box XML.

CHALLENGE

Malicious attacks against networking environments are increasing in frequency and sophistication. To counter these attacks, features are needed that are as flexible as possible, both in terms of classification and mitigation capabilities. Many of the tools available today were not designed with deep packet inspection as a requirement; instead, they were designed to provide matching for pre-defined fields in well-known protocol headers. If an attack uses a field outside the limited range of inspection of these features, one is left without a defense against the attack.

SOLUTION

FPM provides the means to configure match criteria for any or all fields in a packet's header, as well as bit-patterns within the packet's payload within the first 256 bytes. This allows the characteristics of an attack (source port, packet size, byte string) to be uniquely matched and for a designated action to be taken. FPM provides a flexible Layer 2–7 stateless classification mechanism. The user can specify classification criteria based on any protocol and any field of the traffic's protocol stack. Based on the classification result, actions such as drop or log can be taken.

The offset or depth at which to begin matching can be referenced from several locations in the packet. Some of these locations are dependent upon loading a Protocol Header Definition File (PHDF). FPM can work with well-known, established protocols such as IP, TCP, and UDP (PHDFs for these and other protocols are available for download), or with custom protocols that are described with a user-defined PHDF. PHDFs are written in off-box XML.

From the router CLI, FPM is configured using class maps to describe the traffic to be filtered, policy maps to define the action to be taken for filtered traffic, and service policies to attach the filter and action to an interface.

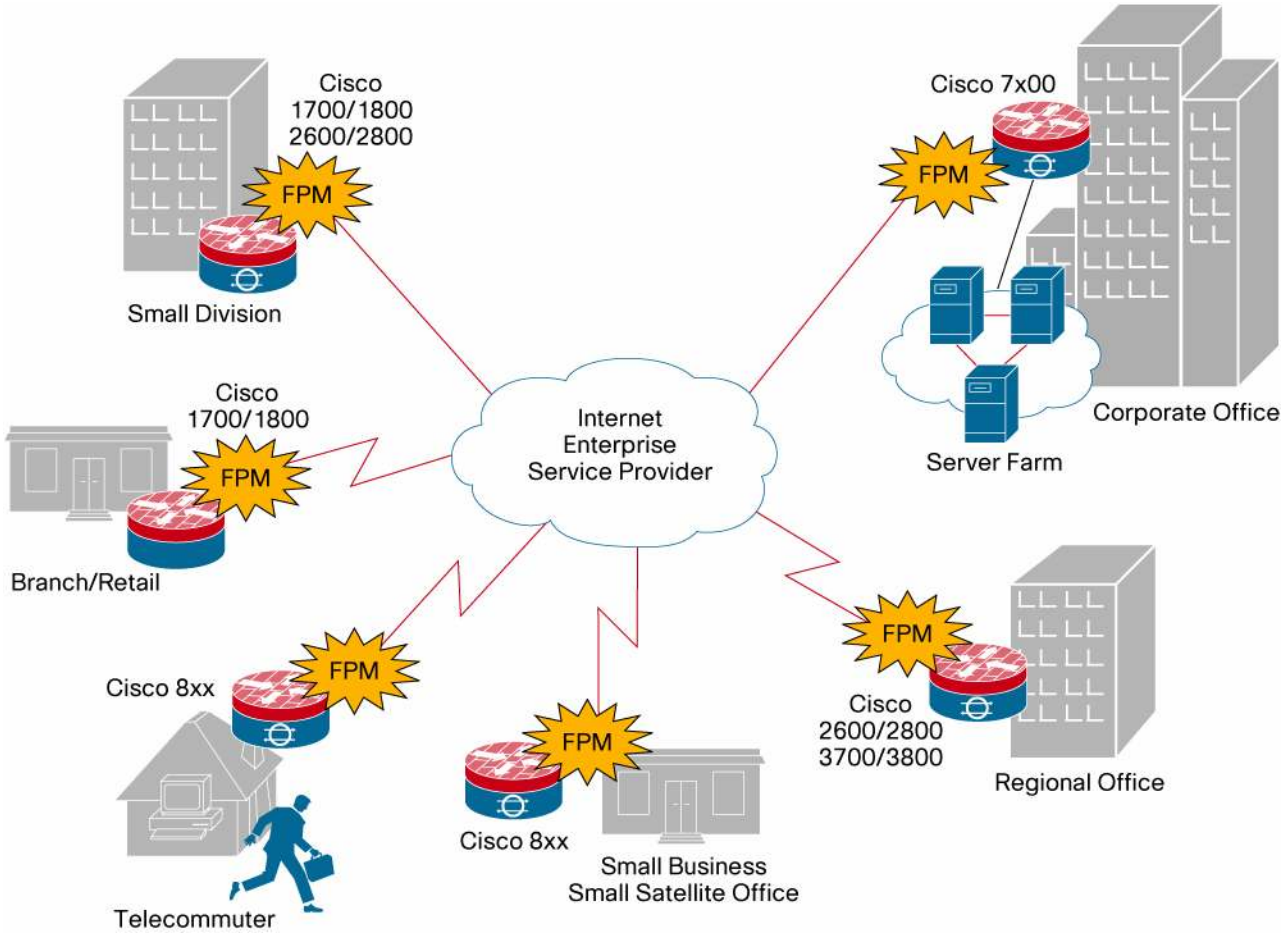
Deployment Scenarios

FPM may be deployed anywhere that the ability to perform classification upon unique bit/byte patterns within IP packets can provide an effective attack mitigation strategy. FPM is not intended to replace an effective IDS/IPS deployment strategy. However, under circumstances where a unique packet classification scheme can be developed, and an IDS/IPS signature is not available (or IDS/IPS is not deployed) and ACLs and/or firewalls cannot provide the appropriate responses, FPM may fulfill the required services.

One example deployment scenario is the use of FPM at the network edge to provide a first line of defense against malicious attacks from outside the network. Figure 1 shows examples of scenarios where FPM deployment would be beneficial.

All contents are Copyright © 1992–2005 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

Figure 1. FPM Deployment Scenarios



Deployment Guidelines

When an attack is suspected in the network, the following steps should be taken to determine whether FPM can be deployed to help mitigate the attack, and to develop the necessary information required to construct an appropriate FPM policy:

Step 1. Determine the characteristics of the attack. Some questions that may help in understanding the nature of the attack include: Does the attack use a specific protocol? Are unique patterns present at specific places within the packets? Does the attack always target a specific port? Are the packets always a specific length?

Step 2. If the results of Step 1 conclude that FPM is useful for mitigating the attack, determine whether existing PHDFs, a custom PHDF, or no PHDFs are required to define the FPM policy. If existing PHDFs are acceptable, skip Step 3 and proceed to Step 4. If a custom PHDF is required, proceed to Step 3 below. If no PHDFs are required (in which case class maps must only use the two permanently defined starting points from the Layer 2 header or the Layer 3 header), skip Steps 3 and 4 and proceed directly to Step 5.

Step 3. Write a custom PHDF for any protocol involved in the attack that is not already covered by an existing PHDF.

Step 4. Load all PHDFs needed to describe the packet contents so that match statements can be written based on convenient PHDF-defined offsets.

Step 5. Configure class maps, policy maps, and services policies to identify the traffic and take an action.

Step 6. Apply the service policies to appropriate interfaces.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

Cisco® Hardware and IOS Software Support

The following table hardware and software that supports FPM.

Table 1. Cisco Hardware and IOS Software Support

Cisco Hardware	Cisco IOS Software Release
Cisco 871	Release 12.4(4)T advipservices
Cisco 1700 Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise, k9o3sy7, bk9no3r2sy7
Cisco 1800 Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise
Cisco 2600/2600XM Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise
Cisco 2800 Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise
Cisco 3700 Series	Release 12.4 (4) T advsecurity, advipservices, adventerprise
Cisco 3800 Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise
Cisco 7200 Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise
Cisco 7300 Series	Release 12.4 (4)T advsecurity, advipservices, adventerprise

Command Syntax

Protocol Header Description File (PHDF)

A PHDF defines each field contained in a particular protocol's header. Each field is described with a name, optional comment, offset, and length. The offset is always specified from the beginning of the header. Both the offset field and the length field may be specified either in terms of bits or bytes.

Standard PHDFs available to be loaded include: ether.phdf, ip.phdf, tcp.phdf, and udp.phdf. These PHDFs provide Layer 2–4 protocol definition. Users may write their own custom PHDFs off-box using the XML language to provide the desired protocol definition through Layer 7.

Note: PHDFs cannot be seen directly within the running config since they are defined using XML and not CLI, but all loaded PHDFs may be displayed using the "show protocols phdf <name>" command. Since PHDFs can only be loaded locally, defined custom PHDFs may also be viewable from local flash memory.

Included in the available PHDFs are several keyword constraints to indicate that the specified field is not just defined as an offset and size, but also that it has a defined (fixed) value. For example, in the PHDF ip.phdf, the IP header length field (called "ihl") is constrained to a value of "5" (5 x 4 bytes = 20 bytes), since variable-length IP headers are not supported at this time. Another example of a keyword constraint refers to the "IP version" field (called "version"), which is constrained to a value of "4" since IP version 6 is not supported at this time. (Note: FPM filters can be constructed without loading PHDFs that allow matches on IP header lengths greater than "5" or IP versions not equal to "4" if these are the desired match criteria. Refer to the next section for constructing FPM filters without loading PHDFs.)

An example XML file for ip.phdf is shown below.

Note: This is a preliminary version.

For reference, the IP header format is included:

IP Header

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				IHL				TOS								Total Length															
Identification																Flags			Fragment Offset											
TTL								Protocol								Header Checksum															
Source IP Address																															
Destination IP Address																															
Options and Padding :::																															

```
<?xml version="1.0" encoding="UTF-8"?>
<phdf>
    <version>1</version>
        <protocol name="ip" description="Definition-for-the-IP-protocol">
                <field name="version" description="IP-version">
                        <offset type="fixed-offset" units="bits">0</offset>
                        <length type="fixed" units="bits">4</length>
                </field>
                <field name="ihl" description="IP-Header-Length">
                        <offset type="fixed-offset" units="bits">4</offset>
                        <length type="fixed" units="bits">4</length>
                </field>
                <field name="tos" description="IP-Type-of-Service">
                        <offset type="fixed-offset" units="bits">8</offset>
                        <length units="bits" type="fixed">8</length>
                </field>
                <field name="length" description="IP-Total-Length">
                        <offset type="fixed-offset" units="bytes">2</offset>
                        <length type="fixed" units="bytes">2</length>
                </field>
                <field name="identification" description="IP-Identification">
                        <offset type="fixed-offset" units="bytes">4</offset>
                        <length type="fixed" units="bytes">2</length>
                </field>
                <field name="flags" description="IP-Fragmentation-Flags">
                        <offset type="fixed-offset" units="bytes">6</offset>
                        <length type="fixed" units="bits">3</length>
                </field>
                <field name="fragment-offset" description="IP-Fragmentation-Offset">
                        <offset type="fixed-offset" units="bits">51</offset>
                        <length type="fixed" units="bits">13</length>
                </field>
                <field name="ttl" description="Definition-for-the-IP-TTL">
                        <offset type="fixed-offset" units="bytes">8</offset>
```

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

```
                <length type="fixed" units="bytes">1</length>
            </field>
            <field name="protocol" description="IP-Protocol">
                <offset type="fixed-offset" units="bytes">9</offset>
                <length type="fixed" units="bytes">1</length>
            </field>
            <field name="checksum" description="IP-Header-Checksum">
                <offset type="fixed-offset" units="bytes">10</offset>
                <length type="fixed" units="bytes">2</length>
            </field>
            <field name="source-addr" description="IP-Source-Address">
                <offset type="fixed-offset" units="bytes">12</offset>
                <length type="fixed" units="bytes">4</length>
            </field>
            <field name="dest-addr" description="IP-Destination-Address">
                <offset type="fixed-offset" units="bytes">16</offset>
                <length type="fixed" units="bytes">4</length>
            </field>
            <field name="payload-start" description="IP-Payload-Start">
                <offset type="fixed-offset" units="bytes">20</offset>
                <length type="fixed" units="bytes">0</length>
            </field>
        <headerlength type="fixed" value="20"></headerlength>
        <constraint field="version" value="4" operator="eq"></constraint>
        <constraint field="ihl" value="5" operator="eq"></constraint>
    </protocol>
</phdf>
```

Filter Description with a Loaded PHDF

Load Required PHDF(s)

Match commands that will require knowledge of fields in the protocol headers of the packet require a PHDF for the protocol(s) to be loaded first.

`rtr(config)# load protocol <location> : <filename.phdf>`

Multiple PHDFs can be loaded. The location of the PHDF(s) must be local to the router.

A specific PHDF can be unloaded, but only if it is not being referenced by any FPM filters. Attempting to unload any PHDFs that are being referenced by a filter will result in an error message, and the PHDF will not be unloaded.

`rtr(config)# no load protocol {<location> : <filename> | <protocol-name>}`

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

Define the FPM Filter Description

A filter description is a definition of a traffic class that can contain the header fields defined in a PHDF. When using a loaded PHDF, the class specification begins with a list of the protocol headers in the packet. For example, a packet might contain IP and TCP headers. A user-defined protocol stack might also include a customized protocol header.

Once the appropriate PHDFs are loaded, a stack of protocol headers must be defined so that FPM knows what headers are present and in what order. This is done using a class-map command with type "stack". If no stack-type class map is specified, the default protocol stack is IP only.

```
rtr(config)# class-map type stack [match-all | match-any] <name>
```

If neither "match-any" nor "match-all" is specified, then it defaults to "match-all".

```
rtr(config-cmap)# match field <loaded protocol name> <protocol field> {eq | neq} <value> [mask <mask value>] next <next protocol>
```

Or

```
rtr(config-cmap)# match field <loaded protocol name> <protocol field> {gt | lt | range | regex } <value>  next  <next protocol>
```

Note: If "range" is specified here, then two values are required—one for the upper limit and the other for the lower limit.

Once the stack of protocols is defined, a class map of type "access-control" is defined for classifying packets.

```
rtr(config)# class-map type access-control [match-all | match-any] <name>
```

The "match-all" option will result in a logical AND of the match statements in this class map. The "match-any" option will result in a logical OR. If not specified, "match-all" is the default.

A description can be added to the class map.

```
rtr(config-cmap)# description <character string>
```

Match statements are now added. Match criteria are defined using a field, offset, operator, value to match, and mask. With a protocol's PHDF loaded, all of the fields defined within the PHDF are available for matching by direct reference to the name of the field. Match definitions can use relational operators, logical operators, string values, and regular expressions.

```
rtr(config-cmap)#match field <loaded-protocol> <protocol field> { eq | neq } <value> [mask <mask value>]
```

Or

```
rtr(config-cmap)#match field <loaded-protocol> <protocol field> { gt | lt  | range | regex } <value>
```

All relational operators (eq, gt, lt, neq) expect the argument to be a number in decimal or hexadecimal. The maximum size of the value is 4 bytes. IP addresses and address masks can be expressed in dotted decimal form or in the decimal or hexadecimal equivalent.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

The optional “mask” is available for “eq” and “neq” operators. The mask value is specified as a “reverse mask”; each bit that you want to examine should be “0” and each “don’t care” bit should be “1”.

If “range” is specified, then two values are required—one for the upper limit and one for the lower limit.

The operator “regex” (regular expression) provides the means to search for a string. The “regex” operator expects a string argument. Hex values may be used in these string arguments. The length is limited by the CLI’s maximum config line length of 255, some of which is used for the command itself. This operator is much more powerful in searches of the payload of the packet that is covered in the next section. While “regex” is available in the match statements of protocol fields, it is very likely that other operators can provide the match function desired with better performance results. The exceptions may be in text-based protocols such as HTTP and with custom protocols.

Operators “eq” and “neq” are preferred over others for performance reasons. If a class map contains multiple match statements, the order of the statements is important. Any “eq” and “neq” statements should be listed first. For a “match-any” class map, processing can stop once any statement is “true”; for a “match-all” class map, processing can stop after any statement is “false”. If the less process-intensive statements are first, then the class map may be resolved before the more process-intensive statements are handled.

Filter Description Without a Loaded PHDF

Define the FPM Filter Description

With no PHDF(s) loaded, there are no fields available to FPM and “match field” is not available. The traffic class must be defined using the datagram (Layer 2) header start or the network (Layer 3) header start. With no PHDF, the “stack” type class map is unnecessary and the classification definition begins with a class map of type “access-control”.

```
rtr(config)# class-map type access-control [match-all | match-any] <name>
```

The “match-all” option will result in a logical AND of the match statements in this class map. The “match-any” option will result in a logical OR. If neither is specified, then the default is “match-all”.

A description can be added to the class map.

```
rtr(config-cmap)# description <character string>
```

With no PHDF loaded, the “field” option of the match statement is not available. Match criteria are defined using a start point, offset, size, value to match and mask. The start points available are the beginning of the Layer 2 datagram header or the Layer 3 network header. Match definitions can use relational operators, logical operators, string values, and regular expressions.

```
rtr(config-cmap)# match start {l2-start | l3-start} offset <value> size <bytes to match> { eq | neq } <value> [ mask < mask value > ]
```

Or

```
rtr(config-cmap)# match start {l2-start | l3-start} offset <value> size <bytes to match> { gt | lt | range | regex } <value>
```

All relational operators (eq, gt, lt, neq) expect the argument to be a number in decimal or hexadecimal. The maximum size of the value is 4 bytes.

The optional “mask” is available for “eq” and “neq” operators. The mask value is specified as a “reverse mask”; each bit that you want to examine should be “0” and each “don’t care” bit should be “1”.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

If “range” is specified here, then two values are required—one for the upper limit and one for the lower limit.

The operator “regex” provides the means to search for a string. The “regex” operator expects a string argument. Hex values may be used in this string argument. Its length is limited by the CLI’s maximum config line length of 255, some of which is used for the command itself. When using this operator, “size” refers to the length of the search block which begins at “offset”. At this time, the upper limit of this block is 32 bytes. This effectively limits the string to be found to 32 bytes maximum as well. The search will begin at the specified offset from the specified start point and move through the search block one byte at a time until the last possible place in the search block where the string could be entirely matched. If the string overlaps the search block boundary, no match will be found.

Regular Expressions

The regular expression support provided will be evolved as needed. Initially, the regular expression support will be minimal. Table 2 lists the characters that are supported.

Table 2. Regular Expression Support

Regular Expression Meta-Character	Function	Example
ASCII Character	Represents the character itself	abcd would match the string “abcd” anywhere in the search area.
.	Represents any character	t..t matches strings such as test, text, and tart
[ ]	A set of characters or a range of characters with (-)	[02468a-z] matches 0, 4, and w, but not 1, 9, or K
*****	Zero or more of preceding characters	5* matches any occurrence of the number 5, including none
?	Zero or one of preceding characters	ba?b matches bb and bab
****	Escape character—specifies what follows as a character instead of a meta-character	18\..* matches the characters 18. and any characters that follow 18.

Policy Map Definition

A policy map is an ordered set of classes and associated actions. The policy binds the class and action. Actions can be drop, icmp response, and log, or service-policy to nest another policy.

```
rtr(config)# policy-map type access-control <name>
rtr(config-pmap)# description <character string>
rtr(config-pmap )# class <class-map name>
rtr(config-pmap-c)# [drop | log | send-response | service-policy]
```

Policy Map Sequence

A new option in the policy map definition allows a new class map to be added at any location in the policy map.

```
rtr(config)# policy-map type access-control <name>
rtr(config-pmap)# class <class-map-name> [insert-before <class-map-name>]
```

If “**insert-before**” keyword is not specified, the new class map is appended to the end of the policy map.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

Service Policy Definition

Service policy is configured on an interface. It can be applied to input and output directions.

The “service-policy” command can be used to call a previously defined policy map.

Finally, the policy is applied to an interface.

```
rtr(config-if)# service-policy type access-control [input | output] <name>
```

Examples Using FPM

1. Fragmented UDP Attack

This attack involves repeatedly sending fragmented UDP packets that tie up the processor, resulting in a denial of service and possibly also resulting in a crash. The characteristics of this attack are UDP packets that have the “more-fragments” flag set or that have a fragment offset greater than zero.

Traditional ACL matches using the “fragments” keyword only recognize packets that include a fragment offset greater than zero. Initial fragments that have a fragment offset equal to zero but that have the “more-fragments” flag set are not matched by the traditional ACL “fragments” keyword.

The PHDFs for IP packets are available for loading. The following commands would be issued in the CLI of the targeted router.

Load the PHDF file for IP since this involves UDP-over-IP packets and will require matching the “more-fragments” flag and the fragment offset, both of which are in the IP header.

```
rtr(config)# load protocol flash:ip.phdf
```

Define the sequence of headers as IP first, then UDP. The “stack” type class map is used for this purpose.

```
rtr(config)# class-map type stack match-all ip_udp
rtr(config-cmap)# description “match UDP over IP packets”
rtr(config-cmap)# match field ip protocol eq 0x11 next udp
```

Define the class map that specifies a match on the “more-fragments” bit OR the “fragment-offset” field. This is done with “match-any”. The IP header fields are shown below:

IP Header

00 01 02 03	04 05 06 07	08 09 10 11	12 13 14 15	16 17 18 19	20 21 22 23 24 25 26 27 28 29 30 31
Version	IHL	TOS		Total Length	
Identification				Flags	Fragment Offset
TTL		Protocol		Header Checksum	
Source IP Address					
Destination IP Address					
Options and Padding :::					

The “more fragments” bit is the third bit of the flags field, so the match statement specifies “eq 1 mask 6”. A mask bit specified as “1” is a “don’t-care”, so a mask of binary “110” (decimal 6) will ensure that bit 3 is the only bit inspected.

When the fragment-offset is greater than zero, the packet is a fragment, so the second match statement specifies this as a characteristic of the class.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

```
rtr(config)# class-map type access-control match-any fragudp
rtr(config-cmap) # description “match on fragmented udp packets”
rtr(config-cmap)# match field ip flags eq 1 mask 6
rtr(config-cmap)# match field ip fragment-offset gt 0
```

Specify the policy map that associates the class defined with an action—in this case, drop.

```
rtr(config)# policy-map type access-control fpm_frag_udp_policy
rtr(config-pmap)# description “policy for fragmented UDP based attacks”
rtr(config-pmap)# class fragudp
rtr(config-pmap-c)# drop
```

Within the final policy definition, first call the “ip_udp” class so that only UDP packets are inspected by the policy defined above. Then, specify the “fpm_frag_udp_policy” policy map to complete the classification and drop action.

```
rtr(config)# policy-map type access-control fpm_policy
rtr(config-pmap)# description “drop fragmented udp packets”
rtr(config-pmap)# class ip_udp
rtr(config-pmap-c)# service-policy fpm_frag_udp_policy
```

Apply the policy to the appropriate interface.

```
rtr(config)# interface GigabitEthernet 0/1
rtr(config-if)# service-policy type access-control input fpm_policy
```

2. Fragmented UDP Attack (Without PHDF Files)

To illustrate the difference between FPM policies when loading PHDFs and when not loading PHDFs, the previous example is repeated below without using the IP PHDF file.

Since there are no PHDFs loaded, the sequence of headers need not be defined via a class map of type “stack”. It is assumed that the packets are IP-only. We can still restrict the class to UDP packets by matching on the IP protocol field in the IP header. However, instead of referring to the “protocol” field as defined in the IP PHDF, in this case we must explicitly define the packet offset and size in order to match the protocol field. Referring to the IP header diagram previously provided, we can see that the protocol field has an offset of 9 bytes from the beginning of the Layer 3 header, and has a size of one byte. In this case, we will be looking for a protocol of 17 (udp).

```
rtr(config)# class-map type access-control udp_pkts
rtr(config-cmap) # description “match on udp packets”
rtr(config-cmap)# match start l3-start offset 9 size 1 eq 17
```

Define the class map that specifies a match on the “more-fragments” bit OR the “fragment-offset” field. The “offset” and “size” parameters are specified in bytes. Therefore, the flags field begins at an offset of 6 bytes within the Layer 3 header, and the “more fragments” bit is the third bit within the flags field. Thus, the match statement specifies “eq 32 mask 0xDF”. A mask bit specified as “1” is a “don’t-care” and a “0” is to be examined, so a mask of binary “1101_1111” (hexadecimal DF) will ensure that the third bit from the left is the only bit inspected. When the “more-fragments” bit is set (value of “1”), the packet is either an initial fragment, or part of a fragmented set of packets with more fragments to follow.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

When the fragment-offset is greater than zero, the packet is a fragment. The second match statement, therefore, specifies this as a characteristic of the class. To match this field, an offset of 6 bytes is also required, however, the mask must specify the inspection of only 13 of 16 bits within a 2-byte field (refer again to the IP header diagram). The mask of binary “1110_0000_0000_0000” (hexadecimal “0xE000”) specifies the first three bits are “don’t care”, providing this capability.

```
rtr(config)# class-map type access-control match-any fragudp
rtr(config-cmap) # description “match on fragmented udp packets”
rtr(config-cmap)# match start l3-start offset 6 size 1 eq 32 mask 0xDF
rtr(config-cmap)# match not start l3-start offset 6 size 2 eq 0 mask 0xE000
```

Specify the policy map that associates the class defined with an action—in this case, drop.

```
rtr(config)# policy-map type access-control fpm_frag_udp_policy
rtr(config-pmap)# description “policy for fragmented UDP based attacks”
rtr(config-pmap)# class fragudp
rtr(config-pmap-c)# drop
```

Within the final policy definition, first call the “udp_pkts” class so that only UDP packets are inspected by the policy defined above.

```
rtr(config)# policy-map type access-control fpm_policy
rtr(config-pmap)# description “drop fragmented udp packets”
rtr(config-pmap)# class udp_pkts
rtr(config-pmap-c)# service-policy fpm_frag_udp_policy
```

Apply the policy to the appropriate interface.

```
rtr(config)# interface GigabitEthernet 0/1
rtr(config-if)# service-policy type access-control input fpm_policy
```

3. TCP SYN Flood

The characteristics of this attack are a flood of TCP SYN packets. TCP connection requests are sent faster than a machine can process them. SYN requests are the first packet in the TCP three-way handshake to establish a TCP connection. Since SYN requests often originate from spoofed addresses, the server under attack is forced to hold the bogus connection request in memory until it times out. The result is denial of service to all requesters—resources are exhausted trying to respond to the flood of requests. If a flood of TCP SYN packets from a spoofed source address is causing a denial of service, then FPM could be used to mitigate the attack. The specific patterns to match would be the SYN flag, and the source address from which the flood of TCP SYN packets is originating.

Traditional ACLs cannot block this attack with the same thoroughness as FPM—they do not have the ability to match directly on TCP flags. Although they do have the “established” keyword (which allows them to permit sessions that are already established), there is no similar mechanism to only block new connection requests from the source address.

The PHDFs for IP and TCP packets are available for loading. The following commands would be issued in the CLI of the router.

Load the PHDF files for the IP and TCP protocols, since this involves TCP-over-IP packets and will require matching the SYN flag in the TCP header and the source IP address in the IP header.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

```
rtr(config)# load protocol flash:ip.phdf
rtr(config)# load protocol flash:tcp.phdf
```

Define the sequence of headers as IP first, then TCP. The “stack” type class map is used for this.

```
rtr(config)# class-map type stack match-all ip_tcp
rtr(config-cmap)# description “match TCP over IP packets”
rtr(config-cmap)# match field ip protocol eq 0x6 next tcp
```

Define a class of traffic where the IP source address is that of the identified attacker (for example, 10.10.10.3) and the TCP SYN flag in the TCP control bits field is set. The TCP header is shown below for reference:

TCP Header

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port	Destination Port
Sequence Number	
Acknowledgment Number	
Data Offset / Reserved / ECN / Control Bits	Window
Checksum	Urgent Pointer
Options and Padding :::	
Data :::	

The TCP control bits field is 6 bits in size and the SYN flag is the fifth bit. The resulting binary mask is ‘11_1101’ (hexadecimal “0x3D”) with a value of decimal “2” when the SYN bit is on. Therefore, the FPM match statement specifies “eq 2 mask 0x3D”; each “1” in the mask indicates “don’t-care”.

```
rtr(config)# class-map type access-control match-all tcpsynflood
rtr(config-cmap) # description “match on tcp syn packets from source address 10.10.10.3”
rtr(config-cmap)# match field ip source-addr eq 10.10.10.3
rtr(config-cmap)# match field tcp control bits eq 2 mask 0x3D
```

Define a policy map to drop the matched traffic and then nest that policy within a policy map matching TCP packets. Finally, apply that service policy to the appropriate interface.

```
rtr(config)# policy-map type access-control fpm_tcp_syn_policy
rtr(config-pmap)# description “policy for TCP SYN flood attacks”
rtr(config-pmap)# class tcpsynflood
rtr(config-pmap-c)# drop

rtr(config)# policy-map type access-control fpm_policy
rtr(config-pmap)# description “drop tcp syn packets from source address 10.10.10.3”
rtr(config-pmap)# class ip_tcp
rtr(config-pmap-c)# service-policy fpm_tcp_syn_policy
```

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

```
rtr(config)# interface GigabitEthernet 0/1
rtr(config-if)# service-policy type access-control input fpm_policy
```

4. HTTP Vulnerability

This attack involves “HTTP Get” packet exploitation. The characteristics of this attack are the use of TCP destination port 80 and the presence of the string “GET \%” somewhere in the first 32 bytes of the TCP payload. Please refer to the following link for more information: http://www.cisco.com/en/US/products/products_security_advisory09186a00800b13c3.shtml

The PHDFs for IP and TCP packets are available for loading. The following commands would be issued in the CLI of the targeted router.

```
rtr(config)# load protocol flash:ip.phdf
rtr(config)# load protocol flash:tcp.phdf
```

First, restrict the number packets that are affected to TCP packets only. We use a class map of type “stack” for this, specifying the IP protocol first, followed by TCP.

```
rtr(config)# class-map type stack match-all ip_tcp
rtr(config-cmap)# description “match TCP over IP packets”
rtr(config-cmap)# match field ip protocol eq 0x6 next tcp
```

Next, define a match for TCP destination port 80 and the presence of the string “GET \%” within the payload.

```
class-map type access-control match-all http_psirt_class
rtr(config-cmap) # description “Match HTTP Exploit Packets”
rtr(config-cmap)# match field tcp dest-port eq 80
rtr(config-cmap)# match start tcp payload-start offset 0 size 32 regex “.*GET \%”

rtr(config)# policy-map type access-control fpm_tcp_policy
rtr(config-pmap)# description “policy for TCP packets”
rtr(config-pmap)# class http_psirt_class
rtr(config-pmap-c)# drop

rtr(config)# policy-map type access-control fpm_policy
rtr(config-pmap)# description “policy HTTP Get exploitation”
rtr(config-pmap)# class ip_tcp
rtr(config-pmap-c)# service-policy fpm_tcp_policy

rtr(config)# interface GigabitEthernet 0/1
rtr(config-if)# service-policy type access-control input fpm_policy
```

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

Monitoring FPM

Show and debug commands can be used to display FPM-related class/policy information.

Show all or designated FPM class map(s).

```
rtr# show class-map type [stack | access-control] [<name>]
```

Show all or designated FPM policy map(s).

```
rtr# show policy-map type access-control [<name>]
```

Show FPM policy map(s) on designated interface. Also show number of packets matched.

```
rtr# show policy-map type access-control interface <interface>
```

Or

```
rtr# show policy-map type access-control control-plane <>
```

Show runtime classification information for the loaded FPM classes and policies.

```
rtr# show protocols phdf <loaded-protocol>
```

Show a listing of user-defined PHDFs stored locally on the router.

```
rtr# dir disk0:*.phdf
```

Track all FPM events in both control plane and data plane.

```
rtr# debug fpm event
```

Performance

Performance tests were run to compare performance of FPM to that of equivalent ACLs. The following table shows processor utilization for these tests. The tests used configurations with 10 FPM classes or equivalent ACLs. Fifty percent of 10 traffic streams generated matches on either the first, fifth, or tenth match statement. In Table 3 below, FPM policies or ACLs with “STD” included in the name use a standard IP source address match. Those that include “EXT” in their name use a 5-Tuple match: IP source, IP destination, TCP source port, TCP destination port, and TCP protocol. Those that include “ALL” in their names use IP source, IP destination, TCP source port range, TCP destination port, and TCP SYN flag matches. For example, “FPM-STD-1-MATCH” configures 10 FPM classes with IP source matches and a corresponding drop action while generating 10 traffic streams to the router with 50 percent of the generated traffic matching the first class.

These tests were run on a Cisco 7206VXR Router with an NPE-400 processor, 128 MB of memory, and a pre-release version of Cisco IOS Software Release 12.4(4)T.

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

Table 3. Processor Utilization for FPM vs. Equivalent ACLs

Filter Type	1000 pps	2000 pps	3000 pps	4000 pps	5000 pps
No Filter	13%	14%	15%	16%	17%
FPM-STD-1-MATCH	16%	33%	49%	64%	70%
ACL-STD-1-MATCH	12%	25%	37%	49%	60%
FPM-STD-5-MATCH	17%	33%	52%	68%	79%
ACL-STD-5-MATCH	12%	25%	37%	48%	60%
FPM-STD-10-MATCH	18%	37%	56%	72%	86%
ACL-STD-10-MATCH	12%	26%	37%	49%	60%
FPM-EXT-1-MATCH	38%	42%	43%	43%	43%
ACL-EXT-1-MATCH	30%	36%	37%	37%	37%
FPM-EXT-5-MATCH	42%	50%	59%	59%	59%
ACL-EXT-5-MATCH	32%	39%	40%	41%	41%
FPM-EXT-10-MATCH	42%	50%	50%	50%	50%
ACL-EXT-10-MATCH	32%	39%	39%	39%	39%
FPM-ALL-1-MATCH	51%	50%	50%	50%	50%
ACL-ALL-1-MATCH	36%	37%	37%	37%	36%
FPM-ALL-5-MATCH	54%	54%	54%	54%	54%
ACL-ALL-5-MATCH	38%	38%	38%	38%	38%
FPM-ALL-10-MATCH	59%	62%	61%	61%	62%
ACL-ALL-10-MATCH	39%	32%	40%	39%	39%

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: 31 0 20 357 1000
Fax: 31 0 20 357 1100

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
168 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on **the Cisco Website at www.cisco.com/go/offices.**

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia • Cyprus Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

Copyright © 2005 Cisco Systems, Inc. All rights reserved. CCSP, CCVP, the Cisco Square Bridge logo, Follow Me Browsing, and StackWise are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, and iQuick Study are service marks of Cisco Systems, Inc.; and Access Registrar, Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StrataView Plus, TeleRouter, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0502R)

205233.CB_ETMG_KS_11.05

Printed in the USA

© 2005 Cisco Systems, Inc. All rights reserved.
Important notices, privacy statements, and trademarks of Cisco Systems, Inc. can be found on cisco.com.