38
Denial of Service (DoS) Protection

Note

- For complete syntax and usage information for the commands used in this chapter, see these publications:

  http://www.cisco.com/en/US/products/ps11846/prod_command_reference_list.html
- Cisco IOS Release 15.4SY supports only Ethernet interfaces. Cisco IOS Release 15.4SY does not support any WAN features or commands.
- Also see:
  - Chapter 34, “MAC Address-Based Traffic Blocking”
  - Chapter 43, “Traffic Storm Control”
  - Chapter 39, “Control Plane Policing (CoPP)”
  - http://www.cisco.com/en/US/docs/ios-xml/ios/security/config_library/15-sy/secdata-15-sy-library.html

Tip

For additional information about Cisco Catalyst 6500 Series Switches (including configuration examples and troubleshooting information), see the documents listed on this page:

http://www.cisco.com/en/US/products/hw/switches/ps708/tsd_products_support_series_home.html

Participate in the Technical Documentation Ideas forum

Security ACLs and VACLs

If the network is under a DoS attack, ACLs can be an efficient method for dropping the DoS packets before they reach the intended target. Use security ACLs if an attack is detected from a particular host.

In this example, the host 10.1.1.10 and all traffic from that host is denied:

```
Router(config)# access-list 101 deny ip host 10.1.1.10 any
Router(config)# access-list 101 permit ip any any
```

Security ACLs also protect against the spoofing of addresses. For example, assume that a source address A is on the inside of a network and a switch interface that is pointing to the Internet. You can apply an inbound ACL on the switch Internet interface that denies all addresses with a source of A (the inside address). This action stops attacks where the attackers spoof inside source addresses. When the packet arrives at the switch interface, it matches on that ACL and drops the packet before it causes damage.

When the switch is used with a Cisco Intrusion Detection Module (CIDM), you can dynamically install the security ACL as a response to the detection of the attack by the sensing engine.

VACLs are a security enforcement tool based on Layer 2, Layer 3, and Layer 4 information. The result of a VACL lookup against a packet can be a permit, a deny, a permit and capture, or a redirect. When you associate a VACL with a particular VLAN, all traffic must be permitted by the VACL before the traffic is allowed into the VLAN. VACLs are enforced in hardware, so there is no performance penalty for applying VACLs to a VLAN.

See Chapter 31, “Cisco IOS ACL Support,” and Chapter 36, “VLAN ACLs (VACLs).”

QoS Rate Limiting

QoS ACLs limit the amount of a particular type of traffic that is processed by the RP. If a DoS attack is initiated against the RP, QoS ACLs can prevent the DoS traffic from reaching the RP data path and congesting it. The PFC and DFCs perform QoS in hardware, which offers an efficient means of limiting DoS traffic (once that traffic has been identified) to protect the switch from impacting the RP.

For example, if the network is experiencing ping-of-death or smurf attacks, the administrator should rate limit the ICMP traffic to counteract the DoS attack and still allow legitimate traffic through the processor, or allow it to be forwarded to the RP or host. This rate limiting configuration must be done for each flow that should be rate limited and the rate-limiting policy action should be applied to the interface.

In the following example, the access-list 101 permits and identifies ping (echo) ICMP messages from any source to any destination as traffic. Within the policy map, a policing rule defines a specified committed information rate (CIR) and burst value (96000 bps and 16000 bps) to rate limit the ping (ICMP) traffic through the chassis. The policy map then is applied to an interface or VLAN. If the ping traffic exceeds the specified rate on the VLAN or interface where the policy map is applied, it is dropped as specified in the markdown map (the markdown map for the normal burst configurations is not shown in the example).

```
Router(config)# access-list 101 permit icmp any any echo
Router(config)# class-map match-any icmp_class
Router(config-cmap)# match access-group 101
Router(config-cmap)# exit
Router(config)# policy-map icmp_policer
Router(config-pmap)# class icmp_class
Router(config-pmap-c)# police 96000 16000 conform-action transmit exceed-action
policed-dscp-transmit drop
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

See Chapter 25, “Classification, Marking, and Policing.”

Global Protocol Packet Policing

Prerequisites for Global Protocol Packet Policing

None.

Restrictions for Global Protocol Packet Policing

- The minimum values supported by the **platform qos protocol arp police** command are too small for use in production networks.
- ARP packets are approximately 40 bytes long and ARP reply packets are approximately 60 bytes long. The policer rate value is in bits per second. The burst value is in bytes per second. Together, an ARP request and reply are approximately 800 bits.
- The configured rate limits are applied separately to the PFC and each DFC. The RP CPU will receive the configured value times the number of forwarding engines.
- Policy-based protocol packet policing is applied per-forwarding engine (PFC and any DFCs).
- With Supervisor Engine 6T, policy-based protocol packet policing supports distributed aggregate policing (see the "Enabling Distributed Aggregate Policing" section on page 25-8).
- The protocol packet policing mechanism effectively protects the RP CPU against attacks such as line-rate ARP attacks, but it polices both routing protocols and ARP packets to the switch and also polices traffic through the switch with less granularity than CoPP.
- The policing mechanism shares the root configuration with a policing-avoidance mechanism. The policing-avoidance mechanism lets the routing protocol and ARP packets flow through the network when they reach a QoS policer. This mechanism can be configured using the **platform qos protocol** *protocol_name* **pass-through** command.
- Policy-based protocol packet policing does not support microflow policers.
- Only ingress Policy-based protocol packet policing is supported.
- Policy-based protocol packet policing does not support Layer 4 ACL operators (see the "Restrictions for Layer 4 Operators in ACLs" section on page 31-2), which imposes these subsequent restrictions:
  - For IPv4 or IPv6 traffic, no support for UDP or TCP port range matching
  - For IPv6 traffic, no support for precedence or DSCP matching
- Protocol packet policing policies can share an aggregate policer with QoS policies.
- An aggregate policer cannot be applied to both ingress and egress traffic.
- With Supervisor Engine 6T, policy-based protocol packet policing supports a maximum of 1,000 global TCAM entries.

- Policy-based protocol packet policing supports the **class default** and **permit** *protocol_name* **any any** commands, but traffic flow might be affected significantly because the protocol packet policing policy processes all matched traffic.
- With Supervisor Engine 6T, policy-based protocol packet policing works with any configured port trust state.
- You can configure both single-command protocol packet policing and policy-based protocol packet policing. Single-command protocol packet policing is applied first and then policy-based protocol packet policing.

Note The software does not detect or attempt to resolve any configuration conflicts between single-command protocol packet policing and policy-based protocol packet policing.

- You can configure both policy-based protocol packet policing and control plane policing (see Chapter 39, "Control Plane Policing (CoPP)"). Policy-based protocol packet policing is applied first and then CoPP.
- Single-command protocol packet policing programs the configured protocol-specific action for ingress traffic and automatically programs a corresponding egress-traffic pass-through action to preserve the ingress result egress traffic.
- Policy-based protocol packet policing does not automatically preserve the ingress policing result in egress traffic.
  - To preserve the ingress policing result in egress traffic with policy-based protocol packet policing, configure an appropriate output policy. To pass egress traffic through unchanged, duplicate each ingess class in the output policy and configure **trust dscp** as the class-map action.
  - Without an output policy-map, egress traffic is processed by any configured interface-based policy-map and ingress global policy result will be overwritten.
- The PFC and any DFCs supports a single **match** command in **class-map match-all** class maps, except that the **match protocol** command can be configured in a class map with the **match dscp** or **match precedence** command.
- The PFC and any DFCs supports multiple **match** commands in **class-map match-any** class maps.
- Class maps can use the **match** commands listed in Table 38-1 to configure a traffic class that is based on the match criteria.

Table 38-1 Traffic Classification Class Map match Commands and Match Criteria

match Commands	Direction	Match Criteria
match access-group {*access_list_number* | **name** *access_list_name*}	Ingress	Access control list (ACL). **Note** Use ACLs to match the following: —CoS value —VLAN ID —Packet length
match any	Ingress	Any match criteria.
match cos	Ingress	CoS value.
match discard-class	Ingress	Discard class value.

Table 38-1 Traffic Classification Class Map match Commands and Match Criteria (continued)

match Commands	Direction	Match Criteria
match dscp **Note** The **match protocol** command can be configured in a class map with the **match dscp** command.	Ingress	DSCP value.
match l2 miss	Ingress	Layer 2 traffic flooded in a VLAN because it is addressed to a currently unlearned MAC-Layer destination address.
match mpls experimental topmost	Ingress	MPLS EXP value in the topmost label.
match precedence **Note** The **match protocol** command can be configured in a class map with the **match precedence** command.	Ingress	IP precedence values.
match protocol {**arp** | **ip** | **ipv6**} **Note** The **match protocol** command can be configured in a class map with the **match dscp** or **match precedence** command.	Ingress	Protocol.
match qos-group	Ingress	QoS group ID.

The PFC and any DFCs supports these ACL types for use with the **match access group** command:

Protocol	Numbered ACLs	Extended ACLs	Named ACLs
IPv4	Yes: 1 to 99 1300 to 1999	Yes: 100 to 199 2000 to 2699	Yes
IPv6	Not applicable	Yes (named)	Yes
MAC Layer	Not applicable	Not applicable	Yes
ARP	Not applicable	Not applicable	Yes

Information About Global Protocol Packet Policing

Attackers may try to overwhelm the RP CPU with routing protocol control packets (for example, ARP packets). Protocol packet policing rate limits this traffic in hardware. Release 15.1(1)SY1 and later releases support policy-based global protocol packet policing, shown in Cisco Feature Navigator as the Global QoS Policy feature.

How to Configure Single-Command Global Protocol Packet Policing

Enter the **platform qos protocol ?** to display the supported routing protocols.

The **platform qos protocol arp police** command rate limits ARP packets. This example shows how to allow 200 ARP requests and replies per second:

```
Router(config)# platform qos protocol arp police 200000 6000
```

This example shows how to display the available protocols to use with protocol packet policing:

```
Router(config)# platform qos protocol ?
  isis
  eigrp
  ldp
  ospf
  rip
  bgp
  ospfv3
  bgpv2
  ripng
  neigh-discover
  wlccp
  arp
```

This example shows how to display the available keywords to use with the **platform qos protocol** command:

```
Router(config)# platform qos protocol protocol_name ?
  pass-through  pass-through keyword
  police        police keyword
  precedence    change ip-precedence(used to map the dscp to cos value)
```

How to Configure Policy-Based Global Protocol Packet Policing

Use these QoS sections and the global protocol packet policing policy map configuration section:

- Configuring a Class Map, page 25-8
- Configuring a Policy Map, page 25-9
- Configuring a Global Protocol Packet Policing Policy Map, page 38-6

Configuring a Global Protocol Packet Policing Policy Map

To configure a global protocol packet policing policy map, perform this task:

Command	Purpose
`Router(config)# platform qos service-policy input policy_map_name`	Configures a global protocol packet policing policy map. **Note** You can configure one input policy.

Unicast Reverse Path Forwarding (uRPF) Check

- Prerequisites for uRPF Check, page 38-7
- Restrictions for uRPF Check, page 38-7
- Information about uRPF Check, page 38-7
- Configuring the Unicast RPF Check Mode, page 38-8
- Enabling Self-Pinging, page 38-9

Prerequisites for uRPF Check

None.

Restrictions for uRPF Check

- Unicast RPF does not provide complete protection against spoofing. Spoofed packets can enter a network through unicast RPF-enabled interfaces if an appropriate return route to the source IP address exists.
- You can configure a unicast RPF mode on each interface.
- The “allow default” options of the unicast RPF modes do not offer significant protection against spoofing.
  - Strict unicast RPF Check with Allow Default—Received IP traffic that is sourced from a prefix that exists in the routing table passes the unicast RPF check if the prefix is reachable through the input interface. If a default route is configured, any IP packet with a source prefix that is not in the routing table passes the unicast RPF check if the ingress interface is a reverse path for the default route.
  - Loose unicast RPF Check with Allow Default—If a default route is configured, any IP packet passes the unicast RPF check.
- Unicast RPF Strict Mode—The unicast RPF strict mode provides the greatest security against spoofed traffic. If, on all unicast RPF-check enabled interfaces, the switch receives valid IP traffic through interfaces that are reverse paths for the traffic, then strict mode is an option.
- Unicast RPF Loose Mode—The unicast RPF loose mode provides less protection than strict mode, but it is an option on switches that receive valid IP traffic on interfaces that are not reverse paths for the traffic. The unicast RPF loose mode verifies that received traffic is sourced from a prefix that exists in the routing table, regardless of the interface on which the traffic arrives.

Information about uRPF Check

The unicast RPF check verifies that the source address of received IP packets is reachable. The unicast RPF check discards IP packets that lack a verifiable IP source prefix (route), which helps mitigate problems that are caused by traffic with malformed or forged (spoofed) IP source addresses.

The PFC4 and DFC4s provide hardware support for the unicast RPF check on up to 16 paths, both with and without ACL filtering, for both IPv4 and IPv6 traffic.

To ensure that no more than 16 reverse-path interfaces exist in the routing table for each prefix, enter the **maximum-paths 16** command in config-router mode when configuring OSPF, EIGRP, or BGP.

How to Configure Unicast RPF Check

Note The following commands exist in the CLI, but have no function:

- **platform ip cef rpf interface-group**
- **platform ip cef rpf multipath interface-group**
- **platform ip cef rpf multipath pass**
- **platform ip cef rpf multipath punt**

Configuring the Unicast RPF Check Mode

To configure unicast RPF check mode, perform this task:

	Command	Purpose
Step 1	`Router(config)# interface {{vlan vlan_ID} | {type slot/port} | {port-channel number}}`	Selects an interface to configure. **Note** Based on the input port, unicast RPF check verifies the best return path before forwarding the packet on to the next destination.
Step 2	`Router(config-if)# ip verify unicast source reachable-via {rx | any} [allow-default] [list]`	Configures the IPv4 unicast RPF check mode.
Step 3	`Router(config-if)# ipv6 verify unicast source reachable-via {rx | any} [allow-default] [list]`	Configures the IPv6 unicast RPF check mode.
Step 4	`Router(config-if)# exit`	Exits interface configuration mode.
Step 5	`Router# show platform hardware cef ip rpf`	Verifies the IPv4 configuration.
Step 6	`Router# show platform hardware cef ipv6 rpf`	Verifies the IPv6 configuration.

- Use the **rx** keyword to enable strict check mode.
- Use the **any** keyword to enable exist-only check mode.
- Use the **allow-default** keyword to allow use of the default route for RPF verification.
- Use the *list* option to identify an access list.
  - If the access list denies network access, denied packets are dropped at the port.
  - If the access list permits network access, packets are forwarded to the destination address. Forwarded packets are counted in the interface statistics.
  - If the access list includes the logging action, information about the packets is sent to the log server.

This example shows how to enable unicast RPF exist-only check mode on Gigabit Ethernet port 4/1:

```
Router(config)# interface gigabitethernet 4/1
Router(config-if)# ipv6 verify unicast source reachable-via any
Router(config-if)# ip verify unicast source reachable-via any
Router(config-if)# end
Router#
```

This example shows how to enable unicast RPF strict check mode on Gigabit Ethernet port 4/2:

```
Router(config)# interface gigabitethernet 4/2
Router(config-if)# ipv6 verify unicast source reachable-via rx
Router(config-if)# ip verify unicast source reachable-via rx
Router(config-if)# end
Router#
```

Enabling Self-Pinging

With unicast RPF check enabled, by default the switch cannot ping itself. To enable self-pinging, perform this task:

	Command	Purpose
Step 1	`Router(config)# interface {{vlan vlan_ID} | {type slot/port} | {port-channel number}}`	Selects the interface to configure.
Step 2	`Router(config-if)# ip verify unicast source reachable-via any allow-self-ping`	Enables the switch to ping itself or a secondary address.
Step 3	`Router(config-if)# exit`	Exits interface configuration mode.

This example shows how to enable self-pinging:

```
Router(config)# interface gigabitethernet 4/1
Router(config-if)# ip verify unicast source reachable-via any allow-self-ping
Router(config-if)# end
```

Configuring Sticky ARP

Sticky ARP prevents MAC address spoofing by ensuring that ARP entries (IP address, MAC address, and source VLAN) do not get overridden. The switch maintains ARP entries in order to forward traffic to end devices or other switches. ARP entries are usually updated periodically or modified when ARP broadcasts are received. During an attack, ARP broadcasts are sent using a spoofed MAC address (with a legitimate IP address) so that the switch learns the legitimate IP address with the spoofed MAC address and begins to forward traffic to that MAC address. With sticky ARP enabled, the switch learns the ARP entries and does not accept modifications received through ARP broadcasts. If you attempt to override the sticky ARP configuration, you will receive an error message.

To configure sticky ARP on a Layer 3 interface, perform this task:

	Command	Purpose
Step 1	`Router(config)# interface type[1] slot/port`	Selects the interface on which sticky ARP is applied.
Step 2	`Router(config-if)# ip sticky-arp`	Enables sticky ARP.
Step 3	`Router(config-if)# ip sticky-arp ignore`	Disables sticky ARP.

1. *type* = **fastethernet**, **gigabitethernet**, or **tengigabitethernet**

This example shows how to enable sticky ARP on interface 5/1:

```
Router# configure terminal
Router(config)# interface gigabitethernet 5/1
Router(config-if)# ip sticky-arp
```

```
Router(config-if)# end
Router#
```

Monitoring Packet Drop Statistics

Prerequisites for Packet Drop Statistics

None.

Restrictions for Packet Drop Statistics

- The incoming captured traffic is not filtered.
- The incoming captured traffic is not rate limited to the capture destination.

Information About Packet Drop Statistics

You can use show commands to display packet drop statistics. You can capture the traffic on an interface and send a copy of this traffic to a traffic analyzer connected to a port, which can aggregate packet drop statistics.

How to Monitor Dropped Packets

Using show Commands

The PFC and DFCs support ACL hit counters in hardware. You can use the **show platform hardware acl entry interface** command to display each entry in the ACL TCAM. You can also use the TTL and IP options counters to monitor the performance of the Layer 3 forwarding engine.

This example shows how to use the **show platform hardware acl entry interface** command to display packet statistics and errors associated with the Layer 3 forwarding engine:

```
Router# show platform hardware statistics

 --- Hardware Statistics for Module 6 ---


L2 Forwarding Engine
```

```
    Switched in L2 : 59624 @ 7 pps

L3 Forwarding Engine
    Processed in L3 : 59624 @ 7 pps
    Switched in L3  : 13 @ 0 pps

    Bridged        : 4602
    FIB Switched
        IPv4 Ucast : 7
        IPv6 Ucast : 1
        EoMPLS     : 1
        MPLS       : 1
        (S , *)    : 0
        IGMP MLD   : 0
        IPv4 Mcast : 2
        IPv6 Mcast : 0
        Mcast Leak : 0
    ACL Routed
        Input      : 1
        Output     : 518
    Netflow Switched
        Input      : 2
        Output     : 0
    Exception Redirected
        Input      : 0
        Output     : 1
    Mcast Bridge Disable & No Redirect
                   : 0
    Total packets with TOS Changed              : 3
    Total packets with TC  Changed              : 0
    Total packets with COS Changed              : 64
    Total packets with EXP Changed              : 0
    Total packets with QOS Tunnel Encap Changed : 1
    Total packets with QOS Tunnel Decap Changed : 1
    Total packets dropped by ACL                : 1
    Total packets dropped by Policing           : 0
Errors
    MAC/IP length inconsistencies               : 0
    Short IP packets received                   : 0
    IP header checksum errors                   : 0
    TTL failures                                : 0
    MTU failures                                : 0

Total packets L3 Processed by all Modules: 59624 @ 7 pps
```

Using SPAN

This example shows how to use the **monitor session** command to capture and forward traffic to an external interface:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# monitor session 1 source vlan 44 both
Router(config)# monitor session 1 destination interface g9/1
Router(config)# end
Router#
```

This example shows how to use the **show monitor session** command to display the destination port:

```
Router# show monitor session 1
Session 1
---------
Source Ports:
```

```
    RX Only:        None
    TX Only:        None
    Both:           None
Source VLANs:
    RX Only:        None
    TX Only:        None
    Both:           44
Destination Ports: Gi9/1
Filter VLANs:       None
```

For more information, see Chapter 18, "Local SPAN, RSPAN, and ERSPAN."

Using VACL Capture

The VACL capture feature allows you to direct traffic to ports configured to forward captured traffic. The capture action sets the capture bit for the forwarded packets so that ports with the capture function enabled can receive the packets. Only forwarded packets can be captured.

You can use VACL capture to assign traffic from each VLAN to a different interface.

VACL capture does not allow you to send one type of traffic, such as HTTP, to one interface and another type of traffic, such as DNS, to another interface. Also, VACL capture granularity is only applicable to traffic switched locally; you cannot preserve the granularity if you direct traffic to a remote switch.

For more information, see Chapter 36, "VLAN ACLs (VACLs)."

Tip For additional information about Cisco Catalyst 6500 Series Switches (including configuration examples and troubleshooting information), see the documents listed on this page:

http://www.cisco.com/en/US/products/hw/switches/ps708/tsd_products_support_series_home.html

Participate in the Technical Documentation Ideas forum