WHITE PAPER

DEPLOYING CONTROL PLANE POLICING

This document will provide an overview of Cisco IOS® Control Plane Policing (CPP), as well as deployment recommendations and guidelines for this Cisco IOS Security Infrastructure feature. CPP is used to increase security on Cisco routers by protecting the Route Processor from unnecessary and often malicious traffic. It allows users to configure a Quality of Service (QoS) filter that manages the traffic flow of control plane packets to protect the control plane of Cisco routers and switches against reconnaissance and Denial of Service (DoS) attacks allowing the Control Plane (CP) to maintain packet forwarding and protocol states despite an attack or heavy load on the router or switch.

PROTECTING THE ROUTE PROCESSOR

A router can be logically divided into four functional components or planes:

1. Data Plane
2. Management Plane
3. Control Plane
4. Services Plane

The vast majority of traffic travels through the router via the data plane; however, the Route Processor must handle certain packets, such as routing updates, keepalives, and network management. This is often referred to as control and management plane traffic.

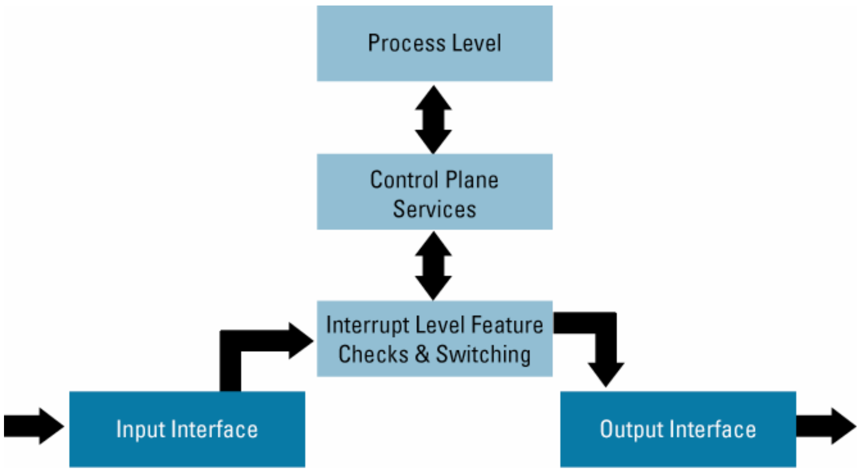
Because the Route Processor is critical to network operations, any service disruption to the Route Processor or the control and management planes can result in business-impacting network outages. A DoS attack targeting the Route Processor, which can be perpetrated either inadvertently or maliciously, typically involves high rates of punted traffic that result in excessive CPU utilization on the Route Processor itself. This type of attack, which can be devastating to network stability and availability, may display the following symptoms:

- High Route Processor CPU utilization (near 100%)
- Loss of line protocol keepalives and routing protocol updates, leading to route flaps and major network transitions
- Interactive sessions via the Command Line Interface (CLI) are slow or completely unresponsive due to high CPU utilization
- Route Processor resource exhaustion—resources such as memory and buffers are unavailable for legitimate IP data packets
- Packet queue backup, which leads to indiscriminate drops (or drops due to lack of buffer resources) of other incoming packets

CPP addresses the need to protect the control and management planes, ensuring routing stability, availability, and packet delivery. It uses a dedicated control-plane configuration via the Modular QoS CLI (MQC) to provide filtering and rate limiting capabilities for control plane packets.

Figure 1 illustrates the flow of packets from various interfaces. Packets destined to the control plane are subject to control plane policy checking, as depicted by the control plane services block.

Figure 1. Packet Flow



COMMAND SYNTAX

CPP leverages MQC to define traffic classification criteria and to specify configurable policy actions for the classified traffic. Traffic of interest must first be identified via class-maps, which are used to define packets for a particular traffic class. Once classified, enforceable policy actions for the identified traffic are created with policy-maps. The `control-plane` global command allows the CP service policies to be attached to control plane itself.

There are four steps required to configure CPP:

1. **Define a packet classification criteria**

```
router(config)#class-map <traffic_class_name>
router(config-cmap)#match <access-group | protocol* | ip prec | ip dscp>
```

2. **Define a service policy**

```
router(config)#policy-map <service_policy_name>
router(config-pmap)#class <traffic_class_name>
router(config-pmap-c)# police <cir | rate> conform-action <transmit | drop > exceed-action <transmit
| drop>
cir        Committed information rate (Bits per second)
rate       Specify policy rate in packets per second (pps)
```

3. **Enter control-plane configuration mode**

```
router(config)#control-plane
```

* When using the 'match protocol' classification criteria, ARP is the only protocol supported. All other protocols need an ACE entry for classification purposes.

4. Apply QoS policy

```
service-policy    {input | output} <service_policy_name>
input             Assign policy-map to the input of an interface
output**          Assign policy-map to the output of an interface
```

COMMAND REFERENCES

Please refer to the following links for more information on the control plane policing and QoS command syntax.

Control Plane Policing: http://www.cisco.com/en/US/products/sw/iosswrel/ps1838/products_feature_guide09186a00801afad4.html#1027184

QoS Command Reference: http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_command_reference_book09186a00801a7ec7.html

CISCO HARDWARE AND CISCO IOS SOFTWARE SUPPORT

Refer to Table 1 for Cisco hardware and Cisco IOS Software support.

Table 1. Cisco Hardware and Cisco IOS Software Support

Cisco Hardware	Cisco IOS Software Release
Cisco 12000 Series Router	Release 12.0(29)S
Cisco 7600 Series	Release 12.2(18)SXD1
Cisco 6500 Series	Release 12.2(18)SXD1
Cisco 7200 Series Cisco 7500 Series	Release 12.2(18)S
Cisco 1751 Router Cisco 2600/2600-XM Series Cisco 3700 Series Cisco 7200 Series	Release 12.3(4)T
Cisco 1800 Series Cisco 2800 Series	Release 12.3(8)T
Cisco 3800 Series	Release 12.3(11)T

DEVELOPING A CPP POLICY

Since CPP filters traffic destined to the Route Processor, it is critical to gain an adequate level of understanding about the legitimate traffic destined for the Route Processor prior to deployment. Configuring CPP policies without this knowledge may result in the blockage of critical traffic, with the potential for unintentionally provoking a DoS attack. In some networks, determining the exact traffic profile required for CPP policies might be difficult, so a careful staged approach should be taken to define these policies. Refer to the Deployment Guidelines section of this document for a recommended conservative methodology for deploying CPP using iterative ACL configurations to help identify and filter traffic.

** Although MQC can be leveraged to support outbound policies, this document focuses solely on input CPP since input CPP provides the most effective protection scenario. Output CPP is mainly used to suppress responses to input packets and does not limit response generation.

Traffic Classification

Prior to developing the actual CPP policy, administrators must identify required traffic and separate it into different classes. Multiple classification schemes can be used, but Cisco recommends a methodology that involves dividing traffic into distinct groups based on relative importance.

The following example uses nine different classes of traffic, thus providing a granular level of detail for real-world environments. Use this as a reference; however, note that the actual number and type of classes needed for a network may differ and should be selected based on local requirements, security policies, and a thorough analysis of the baseline traffic of the customer.

The nine traffic classes in this example were created with the following criteria:

1. **Border Gateway Protocol (BGP)**

- Traffic that is crucial to maintaining neighbor relationships for BGP routing protocol
- Examples: BGP keepalives and routing updates
- Maintaining BGP routing protocol is crucial to maintaining connectivity within a network or to a Service Provider
- Sites that do not run BGP will not need to use this class

2. **Interior Gateway Protocol (IGP)**

- Traffic that is crucial to maintaining IGP routing protocols
- Examples: Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and Routing Information Protocol (RIP)
- Maintaining IGP routing protocols is crucial to maintaining connectivity within a network

3. **Management**

- Necessary, frequently used traffic that is required during day-to-day operations
- Examples: traffic used for remote network access, Cisco IOS Image upgrades and management (ie: telnet, Secure Shell (SSH), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), Terminal Access Controller Access Control System (TACACS), Hypertext Transfer Protocol (HTTP), Trivial File Transfer Protocol (TFTP), and File Transfer Protocol (FTP)

4. **Reporting**

- Traffic used for generating network performance statistics for reporting
- Example: using Cisco IOS IP Service Level Agreements (SLAs) to generate ICMP with different DSCP settings in order to report on response times within different QoS data classes

5. **Monitoring**

- Traffic used for monitoring a router
- Traffic should be permitted but should never pose a risk to the router; with CPP, this traffic can be permitted but limited to a low rate
- Examples: ICMP echo request (ping), and traceroute

6. **Critical Applications**

- Critical application traffic that is specific and crucial to a particular customer environment
- Traffic included in this class should be tailored specifically to the required application requirements of the user (ie: one customer may use multicast, while another uses IPSec and/or Generic Routing Encapsulation [GRE])
- Examples: GRE, Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), Session Initiation Protocol (SIP), Data Link Switching (DLSw), Dynamic Host Configuration Protocol (DHCP), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol (IGMP), Protocol Independent Multicast (PIM), Multicast Traffic, and IPSec

7. **Layer 2 Protocols**

- Traffic used for Address Resolution Protocol (ARP)
- Excessive ARP packets can potentially monopolize Route Processor resources, starving other important processes; CPP can be used to rate limit ARP packets to prevent this
- Currently, ARP is the only Layer 2 protocol that can be specifically classified using the “match protocol” classification criteria

8. **Undesirable**

- Explicitly identifies “bad” or malicious traffic that should be unconditionally dropped and denied access to the Route Processor
- Particularly useful when known traffic destined for the router should always be denied and not placed into a default category; explicitly denying traffic allows the end-user to collect rough statistics on this traffic via show commands and offers some insight into the rate of denied traffic

9. **Default**

- All remaining traffic destined for the Route Processor that has not been identified
- MQC provides the default class, so the user can specify the treatment to be metered out to traffic not explicitly identified in the other user defined classes
- Give this traffic access to the Route Processor at a highly reduced rate
- With a default classification in place, statistics can be monitored to determine the rate of otherwise unidentified traffic destined to the control-plane. Once this traffic is identified, further analysis can be performed to classify it and if needed, the other CPP policy entries can be updated to account for this traffic

Classification Access Lists

Using the classification scheme defined in the previous section, commonly required traffic is identified with a series of Access Control Lists (ACLs). This example uses extended-named ACLs, which are recommended because of their flexibility in allowing the targeted removal and insertion of actions within the ACL. This enables classification ACLs to be updated without completely removing and re-adding them. Named ACLs can also be named using descriptive names to indicate their use.

In this example, the following named classification ACLs are used to classify the traffic into the recommended classes:

- coppacl-bgp: BGP traffic
- coppacl-igp: IGP traffic
- coppacl-management: management traffic
- coppacl-reporting: reporting traffic
- coppacl-monitoring: monitoring traffic
- copp acl-critical-app: critical application traffic
- coppacl-layer2: ARP traffic
- coppacl-undesirable: explicitly denies unwanted traffic (i.e. slammer worm in this example)

The ACLs will be used to build classes of traffic that are used to define the service policies.

Note: As CPP policies are applied to the control plane interface, only traffic destined for the Route Processor will be affected by the CPP policy. The destination key word ‘any’ can be used to classify all traffic destined to any interface on the router. The router can be further secured by specifying exact destination IP addresses, which limits the number of specific interfaces that can receive certain protocols.

In this example, the following named classification ACLs were created for CPP classification:

In this network, BGP is used and must be classified. This ACL only classifies traffic destined to the IP address of 10.9.9.9.

```
ip access-list extended coppacl-bgp
 remark CoPP BGP traffic class
! allow BGP from a known peer to this router’s BGP TCP port
 permit tcp host 47.1.1.1 host 10.9.9.9 eq bgp
! allow BGP from a peer’s BGP port to this router
 permit tcp host 47.1.1.1 eq bgp host 10.9.9.9
 permit tcp host 10.86.183.120 host 10.9.9.9 eq bgp
 permit tcp host 10.86.183.120 eq bgp host 10.9.9.9
```

For the IGP class, OSPF is the IGP used in this example.

```
ip access-list extended coppacl-igp
 remark CoPP IGP traffic class
! permit OSPF
 permit ospf any host 224.0.0.5
 permit ospf any host 224.0.0.6
 permit ospf any any
```

The Management class is for traffic that is required for accessing and managing the router. In this example, TACACS, ssh, telnet, snmp, ntp and FTP (for IOS upgrades) is classified in this class.

```
ip access-list extended coppacl-management
 remark CoPP management traffic class
! permit return traffic from TACACS host
permit tcp host 1.1.1.1 host 10.1.1.1 established
! permit ssh access to the router from a specific subnet
permit tcp 10.86.183.0 0.0.0.255 any eq 22
! permit telnet access to the router from a specific subnet
permit tcp 10.86.183.0 0.0.0.255 any eq telnet
! permit SNMP access from the NMS host to the router
permit udp host 1.1.1.2 any eq snmp
```

Allow the router to receive NTP packets from a known clock source.

```
permit udp host 1.1.1.3 any eq ntp
! Permit FTP from a specific host for IOS upgrades.
permit tcp host 10.9.9.9 eq ftp host 1.1.1.1
permit tcp host 10.9.9.9 eq ftp-data host 1.1.1.1
```

The reporting class is for traffic used for generating network performance statistics. In this example, we are using SAA to generate ICMP Pings with different DSCP bits in order to determine response times for different classes of traffic. i.e. COS1 will use an ICMP with DSCP set to EF, COS2=AF31, COS3=AF21 and COS4=BE. We will create an ACL to classify ICMP pings from specific source routers using SAA to generate the ICMPs. We will then use this ACL and the ‘match ip dscp’ classification criteria in the service policy to create the reporting class.

```
ip access-list extended coppacl-reporting
 remark CoPP reporting traffic class
! permit SAA generated ICMP requests from SAA source router
permit icmp host 10.4.4.4 host 10.1.1.1 echo
```

The monitoring class is used for traffic that is required for monitoring the router. Monitoring traffic is traffic that we expect to see destined to the router and want to track and limit.

```
ip access-list extended coppacl-monitoring
 remark CoPP monitoring traffic class
! permit router originated traceroute
permit icmp any any ttl-exceeded
permit icmp any any port-unreachable
! permit receipt of responses to router originated pings
permit icmp any any echo-reply
! allow pings to router
permit icmp any any echo
```

The critical-app class is used for traffic that is crucial to a specific customer’s environment. In this example, HSRP and DHCP is used.

```
ip access-list extended coppacl-critical-app
 remark CoPP critical apps traffic class
! permit HSRP
permit ip any host 224.0.0.2
! permit DHCP requests
permit udp host 0.0.0.0 host 255.255.255.255 eq bootps
! permit DHCP replies from DHCP server
permit udp host 10.8.8.8 eq bootps any eq bootps
```

This ACL identifies traffic that should always be blocked from accessing the Route Processor. Once undesirable traffic flow is identified, an ACE entry classifying it can be added and mapped to the undesirable traffic class. This can be used as a reaction tool.

```
ip access-list extended coppacl-undesirable
 remark explicitly defined "undesirable" traffic
! permit, for policing, all traffic destined to UDP 1434
permit udp any any eq 1434
```

Note: The coppacl-undesirable ACL is a “permit” entry for classification and monitoring purposes, this traffic will be dropped as a result of the CPP policy.

CPP Policy and MQC

MQC provides a flexible interface for creating service policies. In this configuration, traffic can be identified via the class-map and will be dropped or permitted access to the Route Processor. Using “transmit/drop”, only configuration is analogous to a receive ACL (rACL)*** deployment and when properly deployed provides good protection against many attacks.

MQC also permits multiple match criteria within a class-map. The router needs to determine how packets are evaluated when multiple match criteria exists within a single class. Packets must either meet all of the specified match criteria (**`match-all`**) or any one of the match criteria (**`match-any`**) in order to be considered a member of the class. Traffic destined to the undesirable class should follow a “match-any” classification scheme, traffic that has any of the match criteria specified in this class may be dropped.

The following example shows a typical CPP policy that contains only transmit and drop actions:

The following example shows how to use class-maps to define a class for each “type” of traffic and associate it with an ACL or appropriate match criteria:

```
class-map match-all coppclass-bgp
  match access-group name coppacl-bgp
class-map match-all coppclass-igp
  match access-group name coppacl-igp
class-map match-all coppclass-management
  match access-group name coppacl-management
```

Create a class for packets that match icmp traffic classified by the reporting acl and that are marked with dscp bits of ef, af31, af21 or be.

```
class-map match-all coppclass-reporting
  match access-group name coppacl-reporting
  match ip dscp ef af31 af21 default
class-map match-all coppclass-monitoring
  match access-group name coppacl-monitoring
class-map match-all coppclass-critical-app
  match access-group name coppacl-critical-app
class-map match-all coppclass-layer2
  match protocol arp
class-map match-all coppclass-undesirable
  match access-group name coppacl-undesirable
```

*** For more information regarding rACLs, please visit: http://www.cisco.com/en/US/tech/tk648/tk361/technologies_white_paper09186a00801a0a5e.shtml

The following example illustrates the configuration of the actual CPP service policy. In this base case, all actions are transmit actions. The only exception is for traffic defined as undesirable, which is unconditionally dropped, regardless of rate and the default class that is rate limited above the specified packets per second (PPS) rate. Configuring the CPP policy using only transmit actions initially without the drop action allows the ability to monitor the classes under normal operating conditions in order to determine the appropriate rate limit. Once the appropriate rate under normal conditions is determined, this information can be used to tighten the policy down by dropping packets that exceed the normal operation rates.

Note: The CPP action allows the ability to configure rate limits based on bits-per-second (BPS) or PPS. Rate-limiting at a PPS rate is better as the packet size varies on the network. Understanding the baseline of a network from a PPS perspective is important as well as the bandwidth requirement in BPS.

```
policy-map copp-policy
 class coppclass-bgp
  < no operation specified since this class has unrestricted access to route processor >
 class coppclass-igp
  < no operation specified since this class has unrestricted access to route processor >
 class coppclass-management
     police rate 100 pps conform-action transmit exceed-action transmit
 class coppclass-reporting
     police rate 30 pps conform-action transmit exceed-action transmit
 class coppclass-monitoring
     police rate 50 pps conform-action transmit exceed-action transmit
 class coppclass-critical-app
     police rate 75 pps conform-action transmit exceed-action transmit
 class coppclass-layer2
     police rate 20 pps conform-action transmit exceed-action transmit
```

This policy drops all traffic categorized as undesirable, regardless of rate. The drop command is currently supported only in T train from 12.2(13)T, S train users can use the police statement.

```
 class coppclass-undesirable
     police rate 10 pps conform-action drop exceed-action drop
     <or>
     drop
```

The default class applies to all traffic received by the control plane that has not been otherwise identified. In this example, all default traffic is limited to 10 pps and violations of that limit are dropped.

```
 class class-default
     police rate 10 pps conform-action transmit exceed-action drop
```

Because deploying a policy that permits only required traffic is generally the least complicated option, many administrators will initially configure their network in this manner. In many cases, a simple permit or deny decision will not suffice because it provides only limited flexibility for protecting the Route Processor from those packets that must be allowed with certain restrictions. An example is Internet Control Message Protocol (ICMP) echo-request (pings), which are typically allowed for simple diagnostic purposes. A high enough PPS rate of ping can present a risk, the router must process and generate a reply to each packet. MQC extends the permit/deny model, so echo-request packets can be permitted, but the rate at which those packets are transmitted to the Route Processor is limited. The CPP policy has evolved to the point where traffic can be permitted, denied, or rate limited.

Although rate limiting punted traffic is recommended, care must be taken to ensure that the administrator understands the required rates of traffic, particularly for critical traffic. A very low rate might discard or drop necessary traffic, whereas a very high rate might inundate the Route Processor with a flood of non-critical packets to process. These rates are typically site-specific and vary depending on local topology and routing table size.

Hybrid configurations enable users to deploy a CPP policy that protects the router, while limiting the risk of dropping critical traffic. In a hybrid deployment, some traffic—from known sources and often to specific destinations—is permitted without rate limiting; conversely, different, less critical traffic is policed. The policed rate depends on both determined criticality and site-specific rate values. For instance, the "normal" rate of SNMP will differ based on environment. Using the aforementioned classification scheme with a hybrid deployment, BGP and IGP traffic are both permitted without limitation, while management, reporting, monitoring, critical application, and Layer 2 and default traffic are permitted with appropriate rate limiting. This deployment causes the network to drop undesirable traffic immediately. A hybrid deployment depends on traffic profiling. Using the scheme defined in the previous section, a hybrid configuration involves determining what traffic should pass with a permit statement and how much of that traffic can be permitted (ie: what rate should be allowed). Table 2 extends this example and summarizes a sample policy.

Note: The rates defined in Table 2 are used for illustrative purposes; every environment will have different baselines. For example, a large Service Provider topology will likely require a higher rate of BGP traffic, than a typical enterprise network, due to large BGP routing tables.

The purpose of defining the BGP or IGP traffic class is not to rate limit, but to tag this traffic as BGP or IGP traffic and provide it with unconditional access to the Route Processor. As the policy becomes increasingly refined, a more representative rate should be used for BGP or IGP traffic to further secure the router and show commands that can detect abnormal increases in traffic rates. See the "Monitoring CPP" section of this document for more information regarding CPP centric show commands.

Table 2. Sample CPP Policy

Traffic Class	Rate (PPS)	Conform Action	Exceed Action
BGP	N/A	Transmit	Transmit
IGP	N/A	Transmit	Transmit
Management	250	Transmit	Drop
Reporting	50	Transmit	Drop
Monitoring	75	Transmit	Drop
Critical-App	125	Transmit	Drop
Layer 2	25	Transmit	Drop
Undesirable	10	Drop	Drop
Default	10	Transmit	Drop

The following example shows how to configure the hybrid CPP policy as described in Table 2:

Define a class for each "type" of traffic and associate it with the named classification ACL.

```
class-map match-all coppclass-bgp
  match access-group name coppacl-bgp
class-map match-all coppclass-igp
  match access-group name coppacl-igp
```

```
class-map match-all coppclass-management
  match access-group name coppacl-management
```

Create a class for packets that match icmp traffic classified by the reporting acl and that are marked with dscp bits or ef, af31, af21 or be.

```
class-map match-all coppclass-reporting
  match access-group name coppacl-reporting
  match ip dscp ef af31 af21 default
class-map match-all coppclass-monitoring
  match access-group name coppacl-monitoring
class-map match-all coppclass-critical-app
  match access-group name coppacl-critical-app
class-map match-all coppclass-layer2
  match protocol arp
class-map match-all coppclass-undesirable
  match access-group name coppacl-undesirable
!
```

This is the actual policy. Depending on class of traffic, rates and associated actions are defined.

BGP traffic will not be limited in any way therefore no operation needs to be specified in this class.

Note: Once normal rates are determined for the BGP traffic, consider rate-limiting this class to protect the router further.

```
policy-map copp-policy
  class coppclass-bgp
```

IGP traffic will not be limited in this example either, therefore no operation needs to be specified in this class.

Note: As with the BGP class, once normal rates are determined for the IGP traffic, consider setting a rate-limit to further protect the router.

```
  class coppclass-igp
```

Management traffic is limited to a rate of 250 pps, if traffic exceeds that rate it is dropped.

```
  class coppclass-management
     police rate 250 pps conform-action transmit exceed-action drop
```

Reporting traffic is limited to a rate of 50 pps, if traffic exceeds that rate it is dropped.

```
  class coppclass-reporting
     police rate 50 pps conform-action transmit exceed-action drop
```

Monitoring traffic is limited to a rate of 75 pps, if traffic exceeds that rate it is dropped.

```
class coppclass-monitoring
   police rate 75 pps conform-action transmit exceed-action drop
```

Critical-app traffic is limited to a rate of 125 pps, if traffic exceeds that rate it is dropped.

```
class coppclass-critical-app
   police rate 125 pps conform-action transmit exceed-action drop
```

Layer2 ARP traffic is limited to a rate of 25 pps, if traffic exceeds that rate it is dropped.

```
class coppclass-layer2
   police rate 25 pps conform-action transmit exceed-action drop
```

This policy drops all traffic categorized as undesirable, regardless of rate.

```
class coppclass-undesirable
   police rate 10 pps conform-action drop exceed-action drop
   <or if on the T train you can use the drop command>
   drop
```

The default class applies to all traffic received by the control plane that has not been otherwise identified. In this example, all default traffic is limited to 10 pps and violations of that limit are dropped.

```
class class-default
   police rate 10 pps conform-action transmit exceed-action drop
```

MONITORING CPP

show access-lists

The `show access-lists` command displays a count, per ACL Entry or Access Control Entry (ACE), when traffic matches a particular entry. This information can be used to develop a CPP policy that ensures that identified traffic is matching as expected.

The absence of hits on a particular entry might indicate that the control plane has not received traffic matching the rule. If the lack of traffic matching a specific entry is unlikely, the rule might have been improperly written. It should be carefully reviewed to ensure that the ACL matches the actual traffic.

For instance, in the following example, coppacl-bgp ACL has processed 4 packets from 47.1.1.1 to the BGP port on 10.9.9.9 but no packets from 47.1.1.1's port 179 to 10.9.9.9.

```
server-2821#show access-lists coppacl-bgp
Extended IP access list coppacl-bgp
    10 permit tcp host 47.1.1.1 host 10.9.9.9 eq bgp (4 matches)
    20 permit tcp host 47.1.1.1 eq bgp host 10.9.9.9
    30 permit tcp host 10.86.183.120 host 10.9.9.9 eq bgp (1 match)
    40 permit tcp host 10.86.183.120 eq bgp host 10.9.9.9
```

Displaying an access-list in this manner can also help with attack classification, if an attack is suspected, a show access-list can be used to display an ACE hit count. If one of the entries has logged an excessive count or an unusually rapid rate increase in the number of packets processed, that traffic is suspicious and should be investigated.

In the following output, the router received a significant volume of traffic destined to UDP port 1434. This is clearly attack traffic since UDP/1434 traffic matches the fingerprint of a known worm: Slammer.

```
Extended IP access list coppacl-undesirable
    10 permit udp any any eq 1434 (163053 matches)
```

show policy-map control-plane

The `show policy-map control-plane` command is an invaluable tool for developing site specific policies, monitoring statistics for the control plane policy, and troubleshooting CPP. This command displays dynamic information about the actual policy applied, including rate information and the number of packets (and bytes) that conformed and/or exceeded the configured policies.

The following example, `show policy-map control-plane` output, contains the policy information for three class-maps:

- BGP class: counters display statistics for the number of packets that fall in this class, all 1443 packets are provided access to the Route Processor
- Management class: 262 packets conformed to the defined rate and were forwarded to the Route Processor for processing
- Default class: 6040 packets violated the defined rate of 10 PPS

```
server-2821#sho policy-map control-plane
 Control Plane
  Service-policy input: copp-policy
    Class-map: coppclass-bgp (match-all)
      1443 packets, 113844 bytes
      5 minute offered rate 0 bps
      Match: access-group name coppacl-bgp
.
.
.
    Class-map: coppclass-management (match-all)
      262 packets, 15720 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
```

```
       Match: access-group name coppacl-management
       police:
           rate 250 pps, burst 61 packets
         conformed 262 packets; actions:
           transmit
         exceeded 0 packets; actions:
           drop
         conformed 2 pps, exceed 0 pps
.
.
.
     Class-map: class-default (match-any)
       6335 packets, 385038 bytes
       5 minute offered rate 6000 bps, drop rate 0 bps
       Match: any
       police:
           rate 10 pps, burst 2 packets
         conformed 297 packets; actions:
           transmit
         exceeded 6040 packets; actions:
           drop
         conformed 0 pps, exceed 12 pps
```

MIB SUPPORT

The CBQoSMIB, the primary accounting mechanism for MQC based policies, provides SNMP MIB support for monitoring and managing CPP. This functionality is available, beginning in Cisco IOS Software Releases 12.2(18)S and 12.3(7)T.

For additional information about CBQoSMIB, please visit: http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&mibName=CISCO-CLASS-BASED-QOS-MIB-CAPABILITY

For additional information about the CBQoSMIB configuration and statistics, please visit: ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CLASS-BASED-QOS-MIB.my

OTHER ATTACK VECTORS

Fragmented Packets

Fragmented packets can be used during an attack to provide a form of covertness, to attempt to bypass existing ACLs while increasing the effects of an attack by consuming more resources due to fragmentation reassembly.

ACLs have a fragments keyword that enables specialized handling behavior for fragmented packets.

By default, non-initial fragments (and non-fragmented packets) that match Layer 3 ACL statements, irrespective of the L4 information, are affected by permit and/or deny statements. The use of the fragments keyword modifies this behavior and forces the ACL to either deny or permit non-initial fragments based on Layer 3 information.

In the CPP context, filtering fragments add another layer of protection against a DoS attack with non-initial fragments (i.e. FO > 0). The use of a deny policy for fragments within a CPP policy will deny all subsequent (or non-initial) fragments from accessing the router.

The following ACL will filter all non-initial fragments for each of the protocols listed, and provide basic classification should a fragmentation attack occur, since the access-list entries will log packet counts for each matching packet.

```
 ip access-list extended coppacl-frags
  remark CoPP Fragmented packet traffic class
  permit tcp any any fragments
  permit udp any any fragments
  permit icmp any any fragments
```

The most effective way to apply this ACL is to create a new class, called CPP-class-fragments. The initial policy associated with this class can be a drop policy. This can be changed to a rate limited policy, if necessary. The order of defining classes within a policy is important, as packets will be classified to a particular class based on the order that they are called within a policy. The fragment class must be defined prior to the BGP class, to prevent the non-initial fragments from being classified in any of the other traffic classes. If the fragments class is not defined prior to other classes, fragmented packets that match existing policy information will be handled by that policy.

```
Class-map coppclass-fragments
  Match access-group name coppacl-frags
Class-map coppclass-bgp
  Match access-group name coppacl-bgp
……………
………..
…..

Policy-map copp-policy
  Class coppclass-fragments
    police rate 10 pps conform-action drop exceed-action drop
     <or if the unconditional packet drop command is supported >
    Drop

  class coppclass-bgp

……………..
……..
..
```

Under certain circumstances, a valid session might require fragmentation and it will be filtered if fragments are denied via CPP drop policy. For example, routers acting as IPSec termination devices will often process fragmented packets.

In this case, rate-limiting, rather than drops, should be used:

```
Policy-map copp-policy
  Class coppclass-fragments
     Police rate 10 pps conform-action transmit exceed-action drop
```

For additional information regarding ACL Fragment Processing, visit:
http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800949b8.shtml#intro

Type of Service

Associating a packet with an IP Precedence or IP Differentiated Services Code Point (DSCP) marking allows users to classify traffic based on these values.

- IP DSCP value: first 6 bits in the Type of Service (ToS) byte of an IP packet
- IP Precedence value: first 3 bits in the ToS byte of the IP packet

Cisco routers typically rely on IP precedence for several levels of classification. At a minimum, packets en-queued for Route Processor processing are divided into “normal” and “priority” based on their precedence value. This reliance implies that the precedence values set in packets are trusted and valid.

A CPP policy can be developed to help protect the router from invalid ToS values. Receive path traffic from unknown sources with high ToS values, 6,7 usually reserved for routing protocols, can be dropped, whereas true routing protocol traffic will pass unaffected and therefore be properly classified.

The ToS assurance policy can take on several forms; the simplest and most effective way to drop high precedence traffic from unexpected sources is to define a new traffic classification: ToSDrop. Traffic within this policy will be matched via precedence, rather than by ACL. Traffic will be dropped if precedence matches 6 or 7. The valid ToS 6/7 traffic received from routing peers will be identified and handled by the “BGP” or “IGP” class. For this reason, the ToSDrop class must be defined within the CPP policy only after the classes which contain routing protocols that use ToS6/7 and other classes that rely on ToS6/7 traffic are defined.

The following example shows the CPP ToSDrop class can be integrated into the CPP Policy:

```
class-map match-all coppclass-tosdrop
  match ip precedence 6 7
policy-map copp-policy
  class coppclass-bgp
  class coppclass-igp
  class coppclass-management
     police rate 250 pps conform-action transmit exceed-action drop
  class coppclass-reporting
     police rate 50 pps  conform-action transmit exceed-action drop
  class ToSDrop
     police rate 10 pps conform-action drop exceed-action drop
      < or >
     drop
```

Another option is to define ToS 6 or 7 as matching criteria for the undesirable class. This implies that all traffic received via other classes is trusted and that there is no concern that high precedence attack traffic will fall into another class.

```
Class-map match-any coppclass-undesirable
 Match access-group name coppacl-undesirable
 Match ip precedence 6 7
```

DEPLOYMENT GUIDELINES

By protecting the Route Processor, CPP helps ensure router and network stability during an attack. For this reason, a best practice recommendation is to deploy CPP as a key protection mechanism.

To successfully deploy CPP, the existing control and management plane access requirements must be understood. It can be difficult to determine the exact traffic profile required to build the filtering lists. The following summarizes the recommended steps necessary to properly define a CPP policy:

1. Start the deployment by defining liberal policies that permit most traffic
2. Use show policy-map and show access-lists to determine traffic patterns
3. Use the statistics gathered in step 2 above to tighten policies

The following seven-step guideline describes a very conservative approach for deploying CPP using iterative ACL and policy configurations to help identify and eventually filter traffic:

Step 1. Network Classification Scheme

As described in the earlier section, a classification scheme can simplify the CPP policy development. Enumerate the known protocols that access the Route Processor and divide them into appropriate categories. As described earlier, multiple classification schemes can be used. The recommended classification scheme that was discussed earlier placed traffic in classes called BGP, IGP, Management, Reporting, Monitoring, Critical Applications, Layer 2 Protocols, Undesirable, and Default. The selected scheme must be appropriate for the individual environment, which may require a greater or lesser number of classes.

Step 2. Classification ACLs and Policies

Develop classification ACLs that permit all known protocols that require access to the Route Processor. Cisco recommends the use of distinct named ACLs for each type of traffic defined in step 1 in order to ease management. Each ACL needs to be configured to permit all known protocols in its class that require access to the Route Processor. At this point, each ACL entry should have both source and destination addresses set to **`any`**. In addition, the ACL for the class default should be configured with a single entry, a **`permit ip any any`**. This will match traffic not explicitly permitted by other entries in the other ACLs.

Once the ACLs have been configured, create a **`class-map`** (using descriptive names) for each class defined in step 1, including one for the class default. Next, assign each ACL to its corresponding **`class-map`**. In this step a separate **`class-map`** for the class default should be created, rather than using the **`class-default`** available on some platforms. Creating a separate **`class-map`** and assigning a **`permit ip any any`** ACL will allow the identification of traffic not yet classified, as part of another class to help further refine policy.

The created class maps should be applied to a **`policy-map`** that permits all traffic, regardless of classification. The policy for each class should be set as **`conform-action transmit exceed-action transmit`**.

Step 3. Review Identified Packets and Begin to Filter Access to the Route Processor

Ideally, the classification performed in Step 1 and 2 should identify all required traffic destined to the router; in reality, these steps will not identify all required traffic prior to deployment. The “permit ip any any” access-list entry will log a number of packet matches. Some form of analysis will be required to determine the exact nature of the unclassified packets.

Use the **`show access-lists`** command to determine which entries in the ACL are in use, as well as the number of packets permitted by the final ACL entry (i.e. the match ip any any ACE). The objective is to identify the protocols used by a given network.

Further analyze the unclassified traffic. If necessary use one of the following classification techniques: general ACL classification as described in Characterizing and Tracing Packet Floods Using Cisco Routers****, packet analyzers, or using Receive ACLs (Cisco 7500 Series Router and Cisco 12000 Series Internet Router only)*****. Once traffic has been properly identified, adjust the class configuration accordingly. Remove the ACL entries for those protocols that are not used; and add a permit any entry for each protocol just identified.

Step 4. Restrict a Macro Range of Source Addresses

Refine the classification ACLs by only allowing the full range of the network allocated CIDR block to be permitted as the source address. For example, if the network has been allocated 172.68.0.0/16, then permit source addresses from just 172.68.0.0/16 where applicable.

This step limits risk without breaking services. It also provides data points of devices or people outside the CIDR block, which might be accessing the equipment. External BGP (eBGP) peer will require an exception; the permitted source addresses for the BGP session will lie outside the CIDR block.

This phase might be left on for a few days to collect data for the next phase of narrowing the ACL entries.

Step 5. Narrow the ACL Permit Statements to Only Allow Known Authorized Source Addresses

Increasingly limit the source address to only permit sources that communicate with the Route Processor. An example, only Network Management Stations (NMS) workstations should be permitted to access the SNMP ports on a router. Also only permit telnet or SSH access from addresses assigned to the Network Operations Center (NOC).

Step 6. Refine CPP Policies by Implementing Rate Limiting

Use the **`show`** `policy-map control-plane` command to collect data about the actual policies in place. Analyze the packet count and rate information and develop a rate limiting policy accordingly. Define/adjust the rate limiting policy to allow traffic rates observed under normal conditions plus some headroom for growth while still protecting the router from attacks destined to the router. Enforce the rate limiting policy by configuring the policy action to drop traffic that exceeds the configured rate-limit.

At this point a decision can be made to remove the **`class-map`** that uses the **`permit ip any any`** ACL used for the classification of default traffic. If a decision is made to do this, the previously defined policy for default traffic by the **`class-default`** policy should also be replaced.

Step 7. Limit the Destination Addresses (optional)

Ideally, specific protocols to use specific destination addresses on the router should be allowed. This final phase is designed to limit the range of destination addresses that will accept traffic for a protocol. For instance, BGP peering typically occurs via loopback addresses. Therefore the BGP class can be configured to only allow BGP traffic to the loopback interface.

RISK ASSESSMENT

Care must be taken to ensure that the CPP policy does not filter critical traffic, such as routing protocols or interactive access to the routers. Filtering this traffic could prevent remote access to the router, requiring a console connection. For this reason, lab configurations should closely mimic the actual deployment.

Cisco recommends that customers test this feature in their lab prior to deployment.

SUMMARY

Infrastructure attacks are becoming increasingly common, highlighting the need for infrastructure protection. CPP provides a hardware-independent mechanism for defining and implementing router protection schemes of varying sophistication. It can permit or deny traffic destined to the Router Processor. As operational experience grows, so does the complexity of the policy. The rate limiting features provide flexible policy implementation.

CPP deployment provides several key benefits:

1. Protection against DoS attacks targeted at the network infrastructure
2. Ease of deployment: CPP leverages the existing MQC infrastructure, which allows customers to preserve the existing interface configurations and add global commands to address the aforementioned security goals
3. Consistent implementation strategy across all Cisco hardware
4. Increase the reliability, security, and availability of the network

****	For information on Characterizing and Tracing Packet Floods Using Cisco Routers refer to:
http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a0080149ad6.shtml#topic2

*****	For information on Receive Access Control Lists (RACLs), refer to:
http://www.cisco.com/en/US/tech/tk648/tk361/technologies_white_paper09186a00801a0a5e.shtml