Infrastructure Protection on Cisco IOS Software-Based Platforms

This document describes currently available tools that you can use to protect Cisco IOS software-based infrastructure elements, such as routers and switches, from direct attacks. The same tools can help prevent accidental misconfiguration that may present a risk to the infrastructure. This document also provides deployment guidelines to help facilitate the implementation of these technologies as an integrated security solution, rather than as isolated elements.

The first section provides an overview for the set of basic tools that help mitigate attacks designed to overwhelm the resources available on a device. The next three sections provide a closer look at more advanced features that require additional explanation. The last section provides deployment guidelines explaining how to implement these features in an integrated way. The appendices provide additional useful reference information.

Contents

Cisco Systems

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

Basic Tools and Techniques for Infrastructure Protection

This section describes the following basic tools and techniques, which provide infrastructure protection for Cisco IOS software-based platforms by helping to control the utilization of the limited resources on a device:

- Tuning input hold queues
- Protecting against ICMP unreachable overload
- Configuring Scheduler allocation

In addition to implementing these basic tools and techniques, Cisco IOS software-based devices should be configured according to the device hardening best practices, which help ensure the security of the device by disabling unnecessary services, and by controlling access to the device. Refer to Appendix A for best practices regarding disabling unnecessary services. Refer to Appendix B for best practices regarding device access control.

Tuning Input Hold Queues

The input hold queues hold packets destined to the router or that need to be processed by the route processor (RP). These queues are maintained for each physical interface, and are shared among subinterfaces. With the exception of asynchronous interfaces, the default input queue size is 75 packets, and when that input queue limit is reached the router starts dropping packets.

Denial of service (DoS) attacks against a router can fill the input queue, knocking out legitimate packets. This is especially dangerous for routing and other control plane traffic, such as Border Gateway Protocol (BGP).

Fortunately, the size of the input queue is configurable per interface using the **hold-queue [size]** command from interface configuration mode. Generally speaking, it is recommended to increase the queue to 1500 packets. However, before doing that, it is a good practice to first check the memory available. The number of packets currently set in the input queue can be seen in the "input queue" field in the output from the **show interface** command.

For more information about the **hold-queue** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/inter_r/int_d1g.htm#wp1142192

Protecting Against ICMP Unreachable Overload

According to Internet standards (RFC 1812), whenever a router drops a packet, it should return an ICMP unreachable packet to the packet source. Routers typically drop incoming packets either because they cannot find a valid route or because the packet should be routed to the Null interface. The latter is typically the case with "black hole" filtering. In the past, the Cisco 12000 series routers processed ICMP unreachable packets with the RP, which left an an opening for DoS attacks against the router. It was possible to overwhelm the RP by generating a large amount of packets that required the creation of ICMP unreachables. At this time, ICMP unreachables are handled by the line cards themselves, which protects the RP.

There are two workarounds to solve the issue that affects these older Cisco routers:

- Disable ICMP unreachable messages
- Rate limit ICMP unreachable traffic

The first workaround is to prevent the router from sending ICMP unreachables by entering the **no ip unreachables** command from interface configuration mode as in the following example:

```
router(config)# interface ethernet 0
router(config-if)# no ip unreachables
```

However, in some cases ICMP unreachables are necessary, so preventing the router from sending them is not always appropriate.

The second workaround is to rate limit the number of ICMP unreachables packets that are sent. In Cisco IOS software-based routers this is possible with the **ip icmp rate-limit** command. The Supervisor 720 (Catalyst 6500 Series Switches and the Cisco 7600 Series Routers) provides a hardware-based rate limiter that is configurable with the **mls rate-limit unicast ip icmp unreachable** command.

The following is an example of the **ip icmp rate-limit** command for Cisco IOS software-based routers:

```
router(config)# ip icmp rate-limit unreachable [df] milliseconds
```

Replace *milliseconds* with the number of milliseconds between two consecutive ICMP unreachable packets. The default is 500 ms, which means that no more than one ICMP unreachable packet is sent every 500 ms. The optional **df** flag rate limits ICMP unreachable packets with code 4 (fragmentation required and DF set). The best practice is to set *milliseconds* to 2000 and use the **df** option.

For more information about the **ip icmp rate-limit unreachable** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/ipras_r/ip1_i1g.htm#wp1081902

For more information about the ICMP unreachable rate limiter and other DoS protection controls available on the Supervisor 720, see the following website:

http://www.cisco.com/en/US/partner/products/hw/switches/ps708/products_configuration_guide_chapter09186a0080435872.html

Configuring Scheduler Allocation

Using the **scheduler allocate** command, which schedules CPU time spent on processes versus interrupts, is another good practice to mitigate an ICMP Unreachable overload condition.

When a Cisco router is fast-switching a large number of packets, it can spend so much time responding to interrupts from the network interfaces that no other processing is performed. Some very fast packet floods can cause this condition. The effect can be reduced by using the **scheduler interval** command, which instructs the router to stop handling interrupts and attend to other business at regular intervals. The following is a typical configuration:

```
router(config)# scheduler interval 500
```

This command specifies that process-level tasks will be handled no less frequently than every 500 milliseconds. This command very rarely has any negative effects, and should be a part of your standard router configuration unless you know of a specific reason to leave it out.

Many newer Cisco platforms use the **scheduler allocate** command instead of **scheduler interval**. You use the **scheduler allocate** command to configure two intervals (in microseconds): an interval for the system to run with interrupts enabled, and an interval for the system to run with interrupts masked. If your system does not recognize the **scheduler interval 500** command, try the **scheduler allocate** command, as shown in the following example:

```
router(config)# scheduler allocate 4000 1000
```

The values in this example are those used by the AutoSecure feature, but you should tune these parameters for your specific platform. For more information about the **scheduler allocate** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g06.htm#wp1033741

Infrastructure Protection Access Control Lists

This section describes infrastructure protection access control lists (iACLs), which help prevent or mitigate direct infrastructure attacks by explicitly permitting only authorized traffic to the infrastructure equipment, while allowing transit traffic. Although designed for Internet Service Providers (ISPs), iACLs can also be used to protect the enterprise infrastructure with a few alterations. This section includes the following topics:

- iACL Technology Overview

iACL Technology Overview

Because iACLs are designed as the first line of defense against external threats, they should be deployed at network ingress points. Technically speaking, iACLs consist of extended ACLs that restrict external access to the infrastructure address space. They allow only authorized devices to communicate with infrastructure elements like routers and switches, while letting transit traffic flow freely.

Because they are deployed at network ingress points, it is a good practice to configure iACLs to also provide RFC-1918, RFC-3330, and anti-spoofing filtering.

Note RFC 3330 defines special use addresses that might require filtering. RFC 1918 defines reserved address space that cannot be used for valid source addresses on the Internet. RFC 2827 provides ingress filtering guidelines.

In general, an iACL is composed of the following four sections or modules:

- First module—Special use address and anti-spoofing entries that deny packets from illegitimate sources and that deny packets with source addresses belonging within your address space entering from an external source.
- Second module—Entries providing explicit permission for traffic from legitimate external sources destined to infrastructure addresses.
- Third module—**deny** statements for all other traffic from external sources destined to infrastructure addresses.
- Fourth module—**permit** statements for all other normal backbone traffic en route to non-infrastructure destinations.

Threat Vectors

Use iACLs to help mitigate attacks from external sources directed at infrastructure equipment. Routers, as well as other network devices, have finite processing power, and an excessive amount of traffic directed to them can exhaust their available resources, particularly CPU capacity. When this happens, the router or other network device may start dropping packets, including routing protocol and other control packets. This results in a denial of service (DoS) condition.

Under normal circumstances, most of the traffic handled by a router is data plane traffic, which transits the device over the forwarding path. Only a small portion of the traffic is control and management plane traffic, which is destined to the infrastructure device itself over the receive path for RP handling.

Examples of control and management plane traffic, which is directed to the router itself and which is handled by the RP includes the following:

- Routing protocols
- Remote access protocols, such as SSH and telnet
- Network management traffic, such as SNMP
- Other protocols, such as ICMP, or IP options, might also require processing by the RP

Because iACLs only permit access to the infrastructure IP addresses from authorized sources , they help reduce the scope of attacks targeting the infrastructure from unauthorized external sources.

Functional Overview

You should build iACLs in a structured manner, keeping in mind that ACLs are processed sequentially. The specifics regarding how an iACL should be constructed depend on the particular deployment scenario. However, an iACL generally consists of four distinct sections or modules, which are described in this section:

The first module is designed to block any obvious illegitimate traffic, such as packets arriving with a source IP address belonging to the internal infrastructure address space, as this is clearly spoofing. In addition, the first section of the ACL should also prevent packets with source IP addresses in the private address space (RFC-1918), or special use addresses (RFC-3330). The following is an example of the required entries:

```
! Deny your infrastructure space as a source of external packets
access-list 101 deny ip your_public_infrastructure_block any
! Deny src addresses of 0.0.0.0 and 127/8 (special useIPv4 addresses)
access-list 101 deny ip host 0.0.0.0 any
access-list 101 deny ip 127.0.0.0 0.255.255.255 any
! Deny RFC1918 space from entering AS
access-list 101 deny ip 10.0.0.0 0.255.255.255 any
access-list 101 deny ip 172.16.0.0 0.0.15.255 any
access-list 101 deny ip 192.168.0.0 0.0.255.255 any
```

The second module should allow legitimate routing and management traffic (such as BGP, OSPF, SNMP, or SSH) destined to the infrastructure equipment. This requires a clear understanding of the legitimate traffic required by the specific infrastructure. An iACL built without the proper understanding of the protocols and the devices involved may end up blocking critical traffic. Unless you are careful, the iACL has the potential of *delivering* a DoS attack, instead of preventing it. The section "rACL Deployment Recommendations" section on page 15 describes a technique to facilitate the safe construction of iACLs.

The following configuration fragment shows how the second section of the iACL would look if the only legitimate external traffic were eBGP and OSPF packets from specific peers:

```
! Permit eBGP session
access-list 101 permit tcp host bgp_peer host local_ip eq 179
access-list 101 permit tcp host bgp_peer eq 179 host local_ip
! Permit OSPF
access-list 101 permit ospf host ospf_neighbour host 224.0.0.5
! Permit DR multicast address, if needed
access-list 101 permit ospf host ospf_neighbour host 224.0.0.6
access-list 101 permit ospf host ospf_neighbour host local_ip
```

The third module of the iACL should deny any other external traffic destined to the infrastructure address space, as shown in the following example:

```
! Deny all other access to infrastructure
access-list 101 deny ip any your_public_infrastructure_block
```

The fourth and final module of the iACL should allow any transit traffic. In ISP networks, which are transit networks in nature, this part of the ACL is typically an entry allowing any other IP traffic, such as the following:

```
! Permit transit traffic (ISP).
access-list 101 permit ip any any
```

Contrary to ISP networks, enterprise networks are typically the destination for traffic. Therefore the last section of the iACL can be made more restrictive, by only permitting the protocols and the IP addresses required for legitimate communications on a specific network. For example:

```
! Permit only HTTP service traffic (Enterprise)
access-list 101 permit tcp any 192.168.100.0 eq http
```

This entry allows only HTTP traffic over TCP to a specific network.

Expected Effectiveness

Overall, iACLs are a great help for mitigating direct infrastructure attacks, and are highly recommended. However, there are circumstances in which iACLs may not be effective, including the following:

- Attacks that use the IP addresses and protocols of trusted peers. Attackers may spoof the IP addresses and protocols used by trusted peers to match the permit statements in the iACL.
- DoS initiated from trusted peers. Unintentional attacks may originate from trusted peers as a result of misconfiguration or intentional attacks launched from compromised trusted peers.

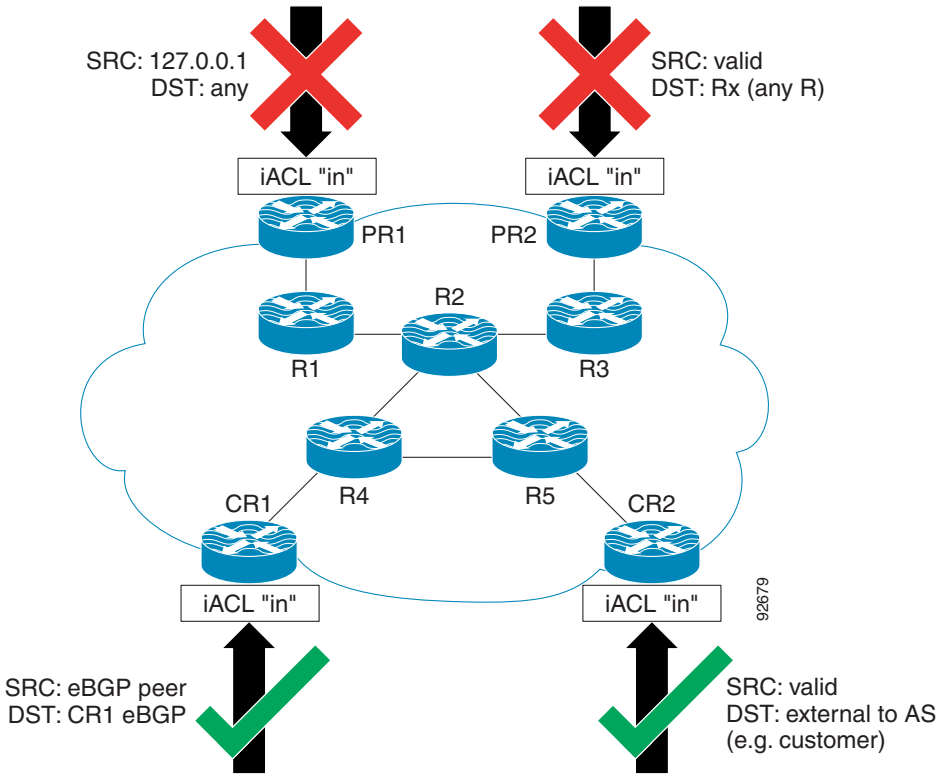
Control Plane Policing (CoPP), explained in the "Control Plane Policing" section on page 21, is a feature that helps mitigate both cases, and its deployment should be considered in conjunction with iACLs.

Also, keep in mind that iACLs are designed to secure the infrastructure and do not provide protection from attacks against targets other than the infrastructure itself.

iACL Deployment Recommendations

The first line of defense against external direct attacks on the infrastructure is provided by iACLs (see Figure 1). For this reason, you should apply iACLs inbound on the ingress interfaces of the routers acting as entry points to the network.

Figure 1 Infrastructure Protection ACLs—The First Line of Defense



As previously mentioned, an iACL built without the proper understanding of the protocols and the devices involved may end up being ineffective and may even result in a denial of service (DoS) condition. For this reason, it is vital to gain an adequate level of understanding about the legitimate traffic destined to the infrastructure before deploying an iACL.

In some networks, determining the exact traffic profile needed to build the filters required might be difficult. For this reason, this document recommends a conservative methodology for deploying iACLs., using iterative ACL configurations to help identify and incrementally filter unwanted traffic as you gain a better understanding of the traffic on your network.

To deploy iACLs using this conservative methodology, complete the following steps:

Step 1 Identify protocols used in the network using a discovery ACL.

Start by deploying a discovery or classification ACL, which permits all the commonly used protocols that access infrastructure devices. "Appendix C—Commonly Used Protocols in the Infrastructure" contains a list of commonly used protocols in the infrastructure.

The discovery ACL should have a source address of **any** and a destination address that encompasses the entire infrastructure IP address space. Logging can be used to develop a list of source addresses that match the protocol **permit** statements. A last line including permitting **ip any any** is required to enable traffic flow.

The objective of configuring the discovery ACL is to determine the protocols that the specific network uses. Use the **log** keyword for analysis to determine what else might be communicating with the router.

Note Although the **log** keyword provides valuable insight into the details of ACL hits, excessive hits to an ACL entry including this keyword might result in an overwhelming number of log entries and possibly high router CPU usage. Only use the **log** keyword for short periods of time as needed to help classify traffic.

Step 2 Review identified packets and begin filtering access to the infrastructure.

Once the packets filtered by the discovery ACL have been identified and reviewed, deploy an ACL with a **permit any** source to infrastructure addresses for the expected protocols.

As in Step 1, the **log** keyword can provide more information about the packets that match the **permit** entries. Using the **deny ip any** *your_public_infrastructure_address_block* command at the end can help identify any unexpected packets destined to the routers. The last line of this ACL should be a **permit ip any any** statement to permit the flow of transit traffic. This ACL will provide basic protection and will allow network engineers to ensure that all required traffic is permitted.

Step 3 Restrict the range of source addresses.

Once you have a clear understanding of the protocols that must be permitted, further filtering can be performed to restrict the protocols to known or authorized source addresses. For example, you can explicitly permit external BGP neighbors or specific GRE peer addresses.

This step narrows the risk of attack without breaking any services and allows you to apply granular control to sources that access your infrastructure equipment.

Step 4 (Optional): Limit the destination addresses within the iACL.

This final phase is meant to limit the range of destination addresses that will accept traffic for a protocol. Some ISPs only allow specific protocols to use specific destination address on the router and this restriction may also be used for enterprise iACLs.

Design Considerations and Limitations

This section describes some additional design considerations and limitations to keep in mind when deploying iACLs. It includes the following topics:

- ACLs and Fragmented Packets
- Understanding Control and Management Plane Traffic
- ACL Performance

ACLs and Fragmented Packets

ACLs provide a **fragments** keyword, which enables specialized fragmented packet handling behavior. In general, non-initial fragments that match the L3 statements (irrespective of the L4 information) in an ACL are affected by the **permit** or **deny** statement of the matched entry.

In the iACL context, filtering fragments adds an additional layer of protection against DoS attacks that use only non-initial fragments (for example, FO > 0). However, the use of a **deny** statement for non-initial fragments at the beginning of the iACL will deny all non-initial fragments from accessing the router. Under rare circumstances, a valid session might require fragmentation and therefore might be filtered if a **deny fragments** statement exists in the iACL.

For example, adding the following entries to the beginning of an iACL would deny any non-initial fragment access to the RP. However, non-fragmented packets or initial fragments would pass to the next lines of the iACL unaffected by the **deny fragments** statements.

```
<<snip>>

access-list 101 deny tcp any your_public_infrastructure_address_block fragments
access-list 101 deny udp any your_public_infrastructure_address_block fragments
access-list 101 deny icmp any your_public_infrastructure_address_block fragments
...
<<snip>>
```

The above rACL snippet also facilitates classification of the attack because each protocol (UDP, TCP, and ICMP) increments separate counters in the ACL.

Because many attacks rely on flooding core routers with fragmented packets, filtering incoming fragments to the core infrastructure provides an added measure of protection and helps ensure that an attack cannot inject fragments by simply matching Layer 3 rules in the iACL.

See the following website for a detailed discussion of the options available for filtering fragments:

http://www.cisco.com/warp/public/105/acl_wp.html#intro

Understanding Control and Management Plane Traffic

Care must be taken to ensure that the iACL does not filter critical traffic such as routing protocols or traffic required for management access to the routers. Filtering required traffic could disable remote access to the router, and connectivity would have to established with a local console connection.

It is recommended that this feature should be tested in a lab configuration that mirrors the actual deployment as closely as possible before deployment.

ACL Performance

ACL performance varies from platform to platform. Before deploying ACLs, review the performance characteristics of your hardware.

Platform and Software Availability

Because iACLs are based on the widely available extended access control lists, they can be universally deployed on any Cisco IOS software-based routers.

iACL Sample Configuration

The example below shows an iACL protecting a router in a scenario involving the following addresses:

- The ISP address block is 169.223.0.0/16
- The ISP infrastructure block is 169.223.252.0/22
- The loopback for the router is 169.223.253.1/32
- The router is a peering router and peers with 169.223.253.2 (to address 169.223.253.1)

The iACL shown below was developed based on this information. The ACL permits external BGP peering to the external peer, provides anti-spoof filters, and protects the infrastructure from all external access.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Infrastructure ACL Example
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
no access-list 110
!
! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Module 1: Anti-spoofing Denies
!--- These ACEs deny fragments, RFC 1918 space,
!--- invalid source addresses, and spoofs of
!--- internal space (space as an external source).
!
!--- Deny fragments.
access-list 110 deny tcp any 169.223.252.0 0.0.252.255 fragments
access-list 110 deny udp any 169.223.252.0 0.0.252.255 fragments
access-list 110 deny icmp any 169.223.252.0 0.0.252.255 fragments
!--- Deny special-use address sources.
!--- See RFC 3330 for additional special-use addresses.
access-list 110 deny ip host 0.0.0.0 any
access-list 110 deny ip 127.0.0.0 0.255.255.255 any
access-list 110 deny ip 192.0.2.0 0.0.0.255 any
access-list 110 deny ip 224.0.0.0 31.255.255.255 any
!--- Filter RFC 1918 space.
access-list 110 deny ip 10.0.0.0 0.255.255.255 any
access-list 110 deny ip 172.16.0.0 0.15.255.255 any
access-list 110 deny ip 192.168.0.0 0.0.255.255 any
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Module 2:  Explicit Permit
!--- Permit only applications/protocols whose destination
!--- address is part of the infrastructure IP block.
!--- The source of the traffic should be known and authorized.
!
!--- Note: This template must be tuned to the network's
!--- specific source address environment. Variables in
!--- the template need to be changed.
!--- Permit external BGP.
access-list 110 permit tcp host 169.223.253.2 host 169.223.253.1 eq bgp
access-list 110 permit tcp host 169.223.253.2 eq bgp host 169.223.253.1
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Module 3:  Explicit Deny to Protect Infrastructure
access-list 110 deny ip any 169.223.252.0 0.0.252.255
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Module 4:  Explicit Permit for Transit Traffic
access-list 110 permit ip any any
```

Useful Debugs and Show Commands

To verify ACL configurations, use the **show** command from EXEC mode.

To display the contents of all access lists, enter the following command:

```
Router# show access-lists
```

To display the contents of one access list, enter the following command:

```
Router# show ip access-list
```

Related Tools

Control Plane Policing (CoPP) complements iACLs by providing protection in the following situations where iACLs are not effective:

- Attacks that spoof the IP addresses and protocols of trusted peers
- DoS attacks initiated from trusted peers

More Information

For information beyond that provided in this section, see *Protecting Your Core: Infrastructure Protection Access Control Lists* (Document ID: 43920), available at the following website:

http://www.cisco.com/en/US/tech/tk648/tk361/technologies_white_paper09186a00801a1a55.shtml

Receive Access Control Lists

Receive Access Controls Lists (rACLs) are designed to protect the RP on high-end routers from unnecessary traffic destined to the RP that could potentially affect system performance.

Originally introduced for the Cisco 12000 Series Routers, rACLs are now available on other high-end routing platforms including the Cisco 7500 Series Routers and the Cisco 10000 Series Routers.

Note Control Plane Policing (CoPP) is preferable, when it is available, because it provides rate limiting in addition to the filtering provided by rACLs (see the "Control Plane Policing" section on page 21).

This section describes how to use and implement rACLs and includes the following topics:

rACL Technology Overview

An rACL is a standard or an extended ACL that controls the traffic sent by the various line cards to the RP on distributed architectures like the Cisco 1200 Series Routers,. Note that rACLs do not apply to transit traffic.

When configured, rACLs are first created on the RP, and then pushed to the line card CPU. When packets enter the line cards, the packets are first sent to the line card CPU. Packets requiring processing by the RP are compared against the rACL before being sent to the RP.

An rACL typically consists of **permit** statements allowing the protocols and sources that are expected by the RP, and may also include **deny** statements explicitly blocking unwanted traffic. Like other ACLs, rACLs have an implicit **deny ip any any** at the end. To activate an rACL, enter the following command from global configuration mode:

```
router(config)# ip receive access-list num
```

Replace *num* with the number of the standard or the extended ACL.

Threat Vectors

An rACL helps mitigate attacks directed at the RP that are intended to overwhelm its capacity. On distributed architectures like the Cisco 12000 Series Router, the RP has a limited capacity to process traffic delivered from the line cards destined to the RP itself. If a high volume of data requires punting traffic to the RP, this can overwhelm the RP, resulting in a denial of service (DoS) condition. The CPU of the RP struggles to keep up with the packet examination and begins dropping packets, thereby flooding the input-hold and Selective Packet Discard (SPD) queues.

Under normal circumstances, most of the traffic handled by a router is in transit over the forwarding path. Only a small portion of the traffic needs to be sent to the RP over the receive path for further analysis . Examples of traffic that is directed to the router itself, and which is handled by the RP includes the following:

- Routing protocols
- Remote access protocols, such as SSH and telnet
- Network management traffic, such as SNMP
- Other protocols, such as ICMP, or IP options, might also require processing by the RP

Because rACLs allow only authorized traffic to be sent to the RP, they help mitigate attacks directed to the RP itself.

Functional Overview

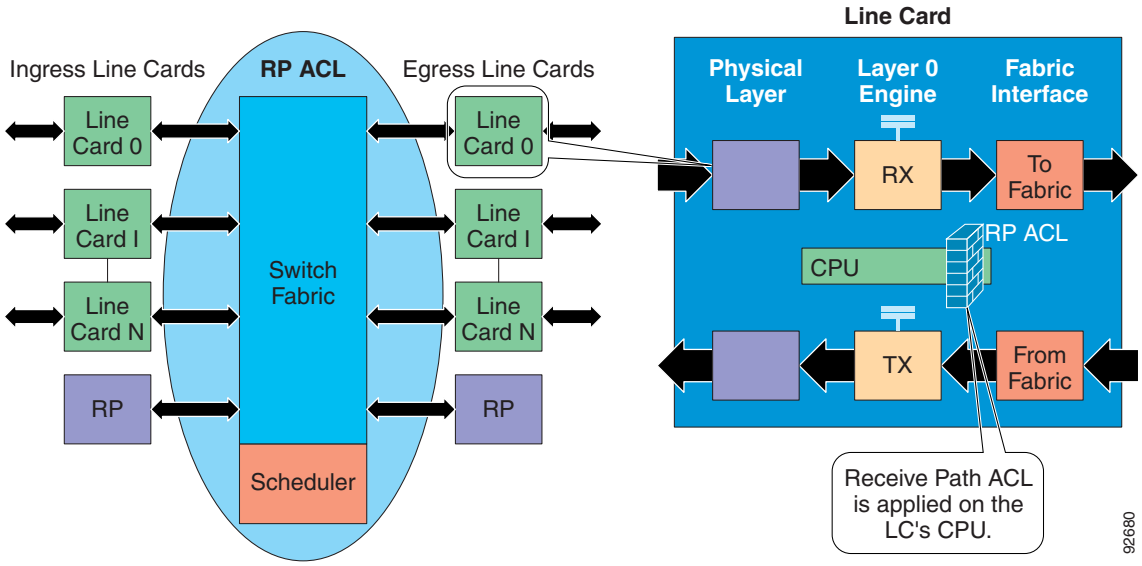
Filtering traffic from the line cards to the RP requires first identifying the packets to be handled by the RP. All traffic with receive adjacencies is subject to rACL filtering. Receive adjacencies are Cisco Express Forwarding (CEF) adjacencies that identify traffic destined to the router IP addresses, such as the broadcast address or the router interface addresses. Filtering based on receive adjacencies works because all traffic handled by the RP has receive adjacencies and most traffic with receive adjacencies is handled by the RP.

There are only a few types of traffic with receive adjacencies that are handled by a line card CPU. The most common packets of this type are ICMP echo requests (pings) that directed to the router, but which are handled by the line card CPU. Because rACLs apply to all packets with receive adjacencies, this type of traffic is also filtered by the rACL even though it is not handled by the RP. Take this into account when designing rACL entries.

When an rACL is configured, it is first created on the RP, and then gets pushed down to the various line card CPUs. Traffic entering a line card is first sent to the local line card CPU, and packets that require processing by the GRP are queued for forwarding to the RP.

Before the packets are sent from the line card CPU to the RP, the traffic is compared against the rACL. If permitted, the traffic passes to the RP; otherwise the traffic is denied. This process is illustrated in Figure 2.

Figure 2 ACLs Protect the Connection Between Line Cards and the Route Processor



<<Note to illustrator: change callout to “Receive path ACL applied on LC CPU” and reduce overall size of diagram>>

An rACL can be constructed by using either a standard or an extended ACL.Once the ACL is defined the rACL can be activated with the **ip receive access-list** *num* global configuration command (*num* is the number of the standard or the extended ACL).

Expected Effectiveness

rACLs help protect the RP from direct attacks by allowing only authorized protocols and sources, and by blocking undesirable traffic. However the effectiveness of rACLs is limited. They either allow or deny traffic and do not provide any rate limiting mechanism that could control large volumes of traffic matching the permitted protocols and sources.

CoPP, which is described in the “Control Plane Policing” section on page 21, provides rate limiting in addition to the filtering capability of rACLs. For this reason, rACLs should only be deployed when CoPP is not available.

rACL Deployment Recommendations

By protecting the RP, rACLs help ensure router and network stability during a DoS attack. For this reason, rACLs (or preferably CoPP if available), should be deployed on all routers as a key protection mechanism.

Because rACLs filter traffic, before deploying rACLs, it is vital to gain an adequate understanding about the legitimate traffic destined to the RP. An rACL built without the proper understanding of the protocols and the devices involved might block critical traffic, potentially creating a denial of service (DoS) condition.

In some networks, determining the exact traffic profile needed to build the filters might be difficult. For this reason, this document recommends a conservative methodology for deploying rACLs using iterative ACL configurations to help identify and eventually filter traffic.

To deploy rACLs using this methodology, complete the following steps:

Step 1 Identify Protocols used in the network with a discovery ACL.

Start by deploying a discovery rACL permitting all the commonly used protocols that access the RP. "Appendix C—Commonly Used Protocols in the Infrastructure" contains a list of commonly used protocols in the infrastructure.

This discovery rACL should have both source and destination addresses set to **any**.

Logging can be used to develop a list of source addresses that match the protocol **permit** statements. In addition to the protocol **permit** statement, a **permit any any log** line at the end of the rACL can be used to identify other protocols that would be filtered by the rACL and might require access to the RP.

The objective is to determine the protocols the specific network uses. Logging should be used for analysis to determine everything else that might be communicating with the router.

Note Although the **log** keyword provides valuable insight into the details of ACL hits, excessive hits to an ACL entry that uses the **log** keyword might result in an overwhelming number of log entries and possibly high router CPU utilization. Use the **log** keyword for short periods of time and only when needed to help classify traffic.

Step 2 Review identified packets and begin to filter access to the RP.

Once the packets filtered by the rACL in Step 1 have been identified and reviewed, implement an rACL with a **permit any any** statement for each of the expected protocols .

As in Step 1, the **log** keyword can provide more information about the packets that match the **permit** entries. Using **deny any any log** at the end can help identify any unexpected packet destined to the RP. This rACL will provide basic protection and allow network engineers to ensure that all required traffic is permitted.

The objective is to test the range of protocols that need to communicate with the router without having the explicit range of IP source and destination addresses.

Step 3 Restrict the range of source addresses.

Only allow addresses within your allocated Classless Inter-Domain Routing (CIDR) block as source addresses. For example, if you have been allocated 171.68.0.0/16 for your network, then permit source addresses only from 171.68.0.0/16.

This step narrows the risk of attack without breaking any services. It also provides data points of devices and users from out side your CIDR block that might be accessing your equipment. Traffic from all outside addresses will be dropped.

An eBGP peer will require an exception because the permitted source addresses for the 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 rACL.

Step 4 Narrow the rACL permit statements to only allow known authorized source addresses.

Increasingly limit the source address to only permit sources that communicate with the RP.

Step 5 (Optional): Limit the destination addresses in the rACL.

Some ISPs only allow specific protocols to use specific destination address on the router and this restriction may also be useful for enterprise rACLs. This final phase is meant to limit the range of destination addresses that will accept traffic for a specific protocol.

Design Considerations and Limitations

This section describes some additional design considerations and limitations to keep in mind when deploying rACLs. It includes the following topics:

- rACLs and Fragmented Packets
- Understanding Control and Management Plane Traffic
- ACL Performance

rACLs and Fragmented Packets

ACLs provide a **fragments** keyword that enables specialized fragmented packet handling behavior. In general, non-initial fragments that match the L3 statements (irrespective of the L4 information) in an ACL are affected by the **permit** or **deny** statement of the matched entry. However, using the **fragments** keyword can force ACLs to either deny or permit non-initial fragments with more granularity.

In the rACL context, filtering fragments adds an additional layer of protection against a DoS attack that uses only non-initial fragments (for example, . FO > 0). However, the use of a **deny** statement for non-initial fragments at the beginning of the rACL prevents all non-initial fragments from accessing the router. Under rare circumstances, a valid session might require fragmentation and therefore be filtered if a **deny fragments** statement exists in the rACL.

For example, adding the following entries to the beginning of a rACL would deny any non-initial fragment access to the RP. However, non-fragmented packets or initial fragments would pass to the next lines of the rACL unaffected by the **deny fragments** statements.

```
access-list 101 deny tcp any any fragments
access-list 101 deny udp any any fragments
access-list 101 deny icmp any any fragments
rest of ACL…
```

This rACL snippet also facilitates classification of the attack because each protocol (UDP, TCP, and ICMP) increments separate counters in the ACL.

See the following website for a detailed discussion of the options available for filtering fragments:

http://www.cisco.com/warp/public/105/acl_wp.html#intro

Understanding Control and Management Plane Traffic

Care must be taken to ensure that the rACL does not filter critical traffic, such as routing protocols or traffic required for management access to the routers. Filtering required traffic could disable remote access to the router, and connectivity would have to established with a local console connection.

It is recommended that this feature be tested before deployment in a lab configuration that mirrors the actual deployment as closely as possible.

rACL Performance Impact

No device memory is consumed by rACLs, other than what is necessary to hold the single configuration entry and the defined ACL itself. The rACL is copied to each line card, so a slight area of memory is taken on each line card. Overall, resources utilized are minuscule, especially when compared with the benefits of deployment.

An rACL does not affect the performance of forwarded traffic. The rACL only applies to receive adjacency traffic, so forwarded traffic is never affected by the rACL. Transit traffic is filtered using interface ACLs, which are applied to interfaces in a specified direction. Traffic is subject to ACL processing before rACL processing, so traffic denied by an interface ACL will not be filtered by the rACL.

The line card that receives the traffic to be filtered by the rACL will have increased CPU utilization due to the processing of the rACL. However, this increased CPU utilization is due to a high volume of traffic destined to the RP so the benefit of protecting the RP far outweighs the increased CPU utilization on the line card. The CPU utilization on a line card will vary based on line card engine type. For instance, given the same attack, an Engine 3-type (E3) line card will have lower CPU utilization than an Engine 0-based (E0) line card.

Enabling turbo ACLs (using the **access-list compiled** command) converts ACLs into a highly efficient series of lookup table entries. When turbo ACLs are enabled, rACL depth does not affect performance. In other words, processing speed is independent of the number of entries in the ACL. However, if the rACL is short, turbo ACLs will not significantly increase performance but will consume memory.

As described above, the rACL is processed on the line card CPU and the CPU utilization on each line card then increases when a large volume of data is directed at the router. On E0, E1 and some E2 bundles, high CPU utilization might lead to routing protocol and link-layer drops. However, these drops are localized to the card and the GRP routing processes are protected, which maintains overall device stability.

E2 cards with throttling-enabled microcode will activate throttling mode when under heavy load and will only forward Precedence 6 and Precedence 7 traffic to the RP. Other engine types have multi-queue architectures. For instance, E3 cards have three separate queues to the CPU, and routing protocol packets (Precedence 6 or 7) are directed to a separate, high-priority queue.

High line card CPU utilization, unless caused by high-precedence packets, will not result in routing protocol drops. Packets to the lower priority queues will be tail-dropped. E4-based cards have eight queues to the CPU, with one dedicated for routing protocol packets.

Platform and Software Availability

rACLs were introduced in 12.0(22)S for the Cisco 12000 series routers. Support for the Cisco 7500 series platform was added in 12.0(24)S, and for the Cisco 10000 Series Routers in 12.3(7)XI1.

rACL Sample Configuration

The following is an example rACL protecting a router in a scenario involving the following addresses:

- The ISP address block is 169.223.0.0/16.
- The ISP infrastructure block is 169.223.252.0/22
- The loopback for the router is 169.223.253.1/32
- The router is a core backbone router, so only iBGP sessions are active.

Given this information, the required rACL could be something like the example shown below. This sample rACL starts with the necessary **deny** statements to block fragments, then continues with a list of explicit **permit** statements that allow the expected management and controls protocols, such as BGP, OSPF, SNMP, and NTP. Finally, the rACL ends with a series of explicit **deny** entries to facilitate the tracking of unexpected traffic destined to the RP.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Receive ACL Example
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
no access-list 110
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Denies any non-initial fragments directed to the RP
access-list 110 deny tcp any any fragments
access-list 110 deny udp any any fragments
access-list 110 deny icmp any any fragments
!
!
! This ACL is an explicit permit ACL. Which means the
! only traffic permitted will be packets which match
! explicit permit ACE.
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Module 1 – Explicit Permit Section
!
! Explicit Permit Phase. Permit only applications whose
! destination address is the loopback and source addresses
! come from an valid host.
!
! Note: This template will need to tuned to the network's
! specific source address environment. Variables in the template
! (like 'bgp_peer') will need to be changed.
!
! Permit BGP
access-list 110 permit tcp 169.223.252.0 0.0.3.255 host 169.223.253.1 eq bgp
access-list 110 permit tcp 169.223.252.0 0.0.3.255 eq bgp host 169.223.253.1
!
! Permit OSPF
access-list 110 permit ospf 169.223.252.0 0.0.3.255 host 224.0.0.5
!
! DR multicast address, if needed
access-list 110 permit ospf 169.223.252.0 0.0.3.255 host 224.0.0.6
access-list 110 permit ospf 169.223.252.0 0.0.3.255 host 169.223.253.1
!
! permit EIGRP
access-list 110 permit eigrp 169.223.252.0 0.0.3.255 host 224.0.0.10
access-list 110 permit eigrp 169.223.252.0 0.0.3.255 host 169.223.253.1
!
! Remote access: telnet and ssh
access-list 110 permit tcp 169.223.252.0 0.0.3.255 host 169.223.253.1 eq 22
access-list 110 permit tcp 169.223.252.0 0.0.3.255 host 169.223.253.1 eq telnet
!
! SNMP
access-list 110 permit udp 169.223.252.0 0.0.3.255 host 169.223.253.1 eq snmp
!
! NTP
access-list 110 permit udp 169.223.252.0 0.0.3.255 host 169.223.253.1 eq ntp
!
! Router originated traceroute
! Each hop returns a ttl exceeded (type 11, code 3) message
! and the final destination returns an ICMP port unreachable
! (type 3, code 0)
```

```
access-list 110 permit icmp any 169.223.253.1 ttl-exceeded
access-list 110 permit icmp any 169.223.253.1 port-unreachable
!
! TACACS for router authentication
access-list 110 permit tcp 169.223.252.0 0.0.3.255 host 169.223.253.1 established
!
! Radius
access-list 110 permit udp 169.223.252.0 0.0.3.255 169.223.253.1 log
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Module 2 – Explicit Denies for Tracking
!
! Deny all other traffic, but (optionally) count denied packets for tracking
access-list 110 deny udp any any !optional can be used to generate denied udp packet
counts
access-list 110 deny tcp any any !optional can be used to generate denied tcp packet
counts
access-list 110 deny icmp any any !optional can be used to generate denied icmp packet
counts
access-list 110 deny ip any any
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Activation of rACL
!
 ip receive access-list 110
```

Useful Debugs and Show Commands

To verify ACL configurations, use the **show command** from EXEC mode.

To display the contents of all access lists, enter the following command:

```
Router# show access-lists
```

To display the contents of one access list, enter the following command:

```
Router# show ip access-list
```

Related Tools

Control Plane Policing (CoPP) is a feature that in addition to the filtering provided by rACL incorporates rate limiting. When available, CoPP is preferable to rACLs.

More Information

For further information about IP rACLs, see the following website:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1829/products_feature_guide09186a00800a8531.html

Control Plane Policing

Control Plane Policing (CoPP) is a security infrastructure feature that allows the configuration of QoS policies to rate limit the traffic sent to the RP in Cisco IOS software-based devices. This protects the control plane from reconnaissance attacks and from direct DoS attacks. This section describes the benefits and functions of different implementations of CoPP and provides implementation instructions and guidelines. It includes the following topics:

CoPP Technology Overview

Packets destined for the RP, called control plane traffic, typically includes the following:

- Routing protocols
- Packets destined to the local IP address of the router
- Packets from network management protocols, such as SNMP
- Interactive access protocols, such as SSH, telnet
- Other protocols, such as ICMP, or IP options, might also require handling by the RP

CoPP regulates control plane traffic by enforcing QoS policies that permit, block, or rate limit traffic to the RP.

CoPP leverages the modular QoS command-line interface (MQC) for configuring QoS policy. MQC allows classification of traffic into classes, and lets you define and apply distinct QoS policies to separately rate limit the traffic in each class. MQC lets you divide the traffic destined to the RP into multiple classes based on different criteria.

For example, four traffic classes could be defined based on relative importance: critical, normal, undesirable and default. The "Developing a CoPP Policy" section on page 28," proposes a more granular and precise class definition.

Note The actual number of classes should be chosen based on local network requirements, security policies, and a thorough analysis of the baseline traffic.

Once the traffic classes are defined, a specific QoS policy can be defined and enforced for each class according to importance. The criteria for defining the traffic classes can be configured using various mechanisms, including extended ACLs and precedence matching.

MQC allows the definition of a flexible QoS policy that applies one of the following system actions to a specific traffic class:

- Permit all packets
- Drop all packets
- Drop packets exceeding the specified rate limit

This flexibility makes CoPP more effective than rACLs, which only permit or deny traffic, but cannot rate limit. There are scenarios in which a binary response of permit or deny is not sufficient.

Within the traffic you want to permit, there may be specific types of traffic that require different treatment. For example, traffic belonging to the critical and normal classes should all be permitted, but the critical traffic should be given higher precedence.

There are other situations where traffic should be allowed subject to certain restrictions. For example, ICMP echo requests (pings) are commonly allowed for diagnostic purposes and should be permitted when used for their intended purpose. However, a large volume of ICMP echo-requests can overwhelm the RP and may be part of a DoS attack.

CoPP can effectively handle this kind of situation by enforcing rate limiting policies per traffic class. For example, using MQC, you can define a traffic class to include ICMP echo requests and then drop packets exceeding the specified rate limit for this traffic class.

Threat Vectors

CoPP helps protect the RP of Cisco IOS software devices in more than one way. From a policing perspective, by filtering traffic sent to the RP, CoPP ensures that only the expected protocols are allowed. This effectively shields the control plane from unwanted and potentially malicious traffic. On the other hand, by rate limiting the traffic sent to the RP, CoPP provides protection against large volumes of packets that might be part of a DoS attacks. This helps maintain network stability even during an attack.

From a packet classification perspective, CoPP also helps mitigate attacks based on fragments or on invalid Type of Service (TOS) values. Fragments are typically used in attacks to bypass ACLs or to overwhelm a target with the processing required to reassemble fragments. CoPP can control fragments by classifying traffic with extended ACLs using the **fragments** keyword. In this way, a specific class of traffic can be defined for fragments, and the MQC policy can either block or rate limit packets in this traffic class.

Invalid TOS values can also be used in attacks. Often routers and switches rely on IP Precedence or Differentiated Services Code Point (DSCP) marking to prioritize traffic for queuing or processing purposes. An attacker can try to use higher priority values to either bypass security or to give attack traffic higher precedence. CoPP can use a combination of extended ACLs and precedence matching to identify traffic from untrusted sources with high TOS values. This traffic can be associated with a class and an MQC policy can be applied to drop all packets in the class.

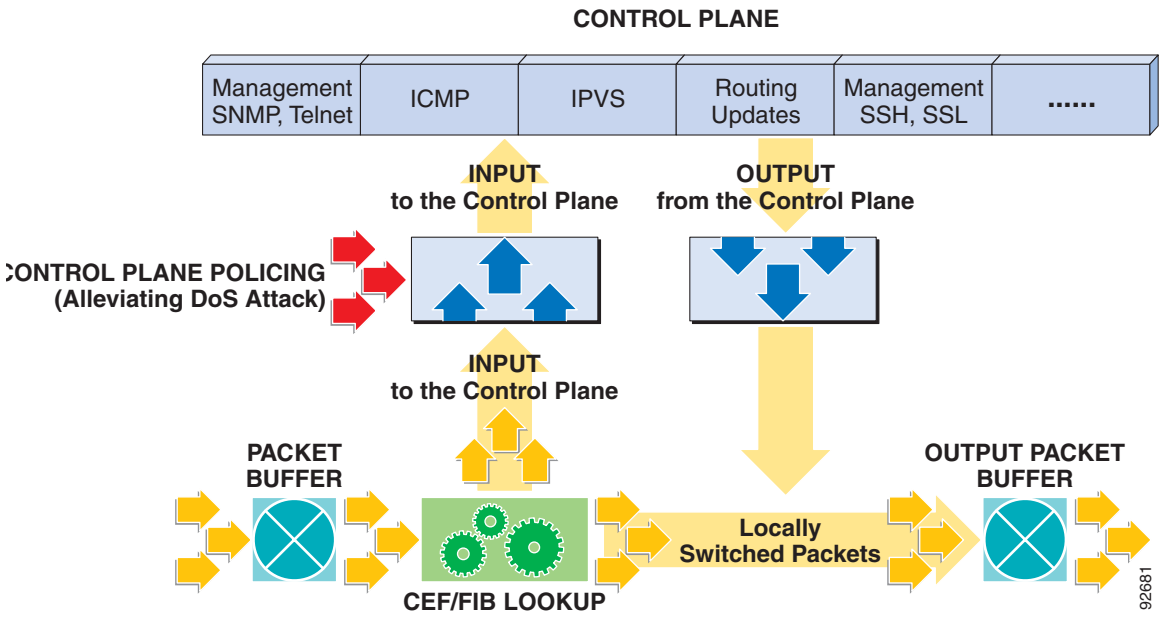
Functional Overview

CoPP is available on a wide range of Cisco IOS software-based platforms, which all deliver the same basic functionality. However, CoPP has been enhanced on some platforms to leverage the benefits of their hardware architectures. As a result, some platforms provide more advanced forms of CoPP.

Non-distributed platforms implement a centralized CoPP model, while some distributed platforms provide enhanced versions of CoPP: distributed and hardware-based. This section describes the existing CoPP models and their functional characteristics and includes the following topics:

- Centralized CoPP
- Cisco 1200 Series Routers Distributed CoPP
- Supervisor 720 (Catalyst 6500 and Cisco 7600) CoPP

Figure 3 Centralized CoPP



Centralized CoPP

Centralized CoPP is implemented on non-distributed platforms, and is also available on some distributed platforms. The centralized model implements CoPP as a single entity.

As illustrated in Figure 3, CoPP comes into play right after the switching or the routing decision, and before traffic is forwarded to the control plane. The sequence of events, at a high level, is as follows:

1. A packet enters a router or a switch configured with CoPP on the ingress interface.
2. The interface performs the basic input port and QoS services.
3. The packet gets forwarded to the router or switch engine.
4. The router or switch engine makes a routing or a switching decision, determining whether or not the packet is destined to the control plane.
5. Packets destined for the control plane are processed by CoPP, and are dropped or delivered to the control plane according to the polices for each traffic class.

   Packets that have other destinations are forwarded normally.

As mentioned earlier, the CoPP policies that define how control plane packets are handled are configured using MQC. The first two steps are to define a traffic classification scheme and to apply policy actions (forward, drop, or rate limit), for each class. Traffic can be classified in many ways, but generally it is a good idea to separate traffic based on relative importance. For example, traffic could be classified into four classes: critical, normal, undesirable, and default.

Note The actual number of classes should be chosen based on local network requirements, security policies, and a thorough analysis of the baseline traffic.

To enable CoPP, first define and classify the control plane traffic using the **class-map** command to define each class. The syntax for this command is as follows:

```
router(config)# class-map traffic_class_name
```

The **class-map** command defines the class map name and enables a configuration mode for defining the class. Within the traffic class configuration mode, use the **match** command to associate specific traffic with the class. The syntax for this command is as follows:

```
router(config-cmap)# match access-group group-name| protocol arp | ip prec | ip dscp
```

Once the traffic is classified, you apply a policy action to each class, indicating whether to permit all packets, to drop all packets, or to drop packets crossing a specified rate limit for that particular class. To apply these policy actions use the **policy-map** command, which has the following syntax:

```
router(config-pmap)# policy-map service_policy_name
```

The **policy-map** command defines the policy map name and enables a configuration mode for defining the policy. You then use the **class** command to associate one or more traffic classes with the policy. You use the **police** command to define the policy action to apply. The syntax for these commands is as follows:

```
router(config-pmap)# class traffic_class_name
router(config-pmap-c)# police bps-rate [burst-normal] [burst-max] conform-action transmit
exceed-action drop
```

Finally, apply the defined CoPP policy to the control plane by using the **service-policy** command from **control-plane** configuration mode.

```
router(config)# control-plane
router(config-<<???>>)# service-policy {input | output} service_policy_name
```

Use the **input** option to assign the policy map to the traffic received on an interface. Use the **output** option to assign the policy map to the traffic transmitted on an interface.

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

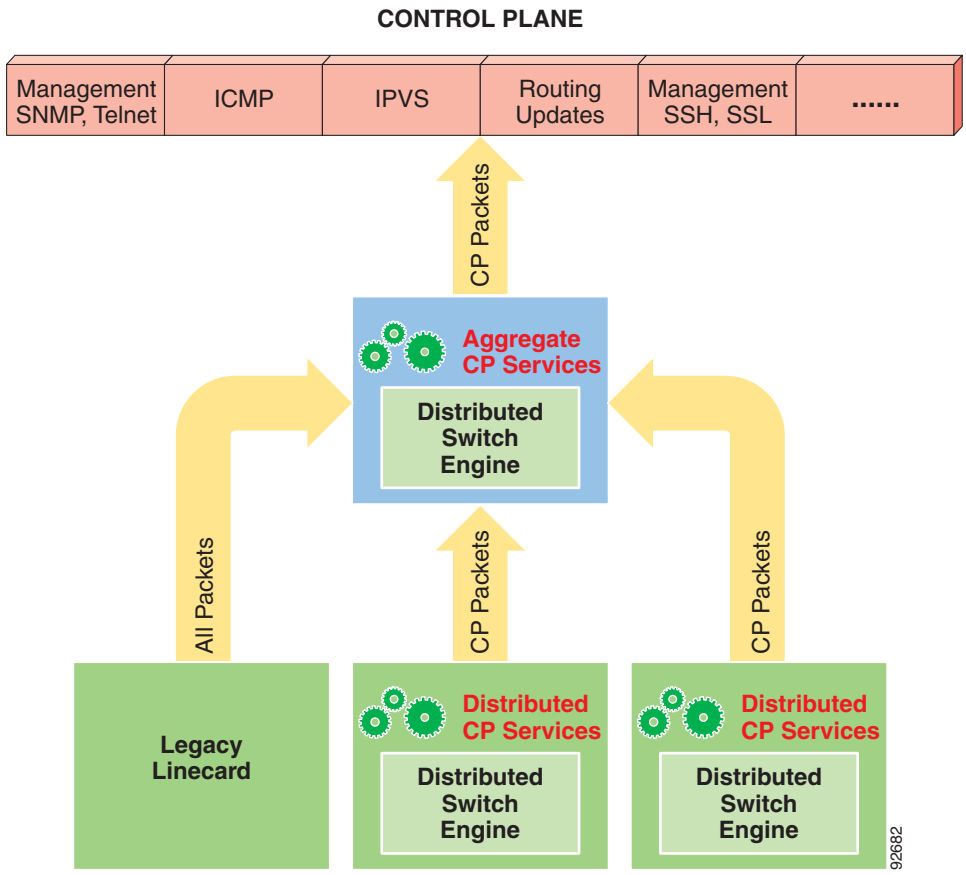
Cisco 1200 Series Routers Distributed CoPP

The Cisco 12000 Series Routers implement a distributed form of CoPP, called Distributed CoPP. Distributed CoPP takes advantage of the processing power present on some line cards by running a CoPP instance on each line card. Distributed CoPP implements a centralized instance of CoPP at the RP level, which processes all control packets arriving from all router interfaces. However, Distributed CoPP also allows the definition of a CoPP instance per line card that processes only the packets arriving from the line card interfaces. In this way, Distributed CoPP provides two layers of control:

- Distributed Control Plane Services at the line card level
- Aggregated Control Plane Services at the RP level

These two layers of control give Distributed CoPP greater flexibility and granularity. First, when packets are received they are processed right at the interface level by the distributed control plane component. Then, a second layer of protection is applied to all packets just before getting onto the control plane. This is illustrated in Figure 4.

Figure 4 Distributed CoPP



The sequence of events for the first layer of control is as follows:

1. A packet arrives at the router over a distributed line card with processing power.
2. The packet is processed by the line card distributed switch engine, which performs basic port and QoS services.
3. The line card switch engine makes a routing decision, determining whether or not the packet is destined to the control plane.
4. In case the packet is destined to the control plane, the packet is processed by the Distributed Control Plane Service in the same line card.
5. Depending on the CoPP policy, the packet is dropped or delivered to the central switch engine for further processing.

If the packet is sent to the central switch engine, the packet gets processed by the Aggregate Control Plane Service. According to the aggregate CoPP policy, the packet is then dropped or delivered to the control plane.

Packets reaching the router over non-distributed line cards (legacy line cards) are only processed at the aggregate layer. These packets are forwarded directly to the central switch engine for a routing decision. Packets destined to the control plane are handled by the Aggregate Control Plane Services, which drop or deliver the packets according to the aggregate CoPP policy.

Distributed CoPP has multiple benefits over centralized (non-distributed) CoPP. Distributed CoPP provides more granular and more effective policy definition. Different line cards may connect to networks with distinct security needs. With distributed CoPP each line card can be configured with a unique CoPP policy that controls the traffic coming from all ports in the line card, and that best accommodates its particular security needs. For example, BGP filtering could be defined on the line cards that use BGP, while the rest of the filtering requirements can be handled by the aggregate services.

At the same time, distributed CoPP policies can be configured to offload processing from the central switch engine. Distributed CoPP also provides more effective policy enforcement. With non-distributed CoPP, there is no effective way to prevent one interface from consuming all the aggregate control plane resources. A DoS attack on one port may negatively impact control plane processing of the traffic from other ports. Distributed CoPP can effectively handle this situation by controlling the amount of traffic sent by each line card to the control plane.

Distributed CoPP is configured at two levels: at the aggregate level, and optionally at the distributed line cards. Distributed control plane services are optional, but they provide more refined policy definition and enforcement and are highly recommended. The configuration of both aggregate and distributed services first requires the classification of the traffic using the **class-map** command, and then assigning policy actions to each traffic class with the **policy-map** command. This configuration is similar to configuring centralized CoPP policy. However, in this case, multiple policies can be defined: one for Aggregate Control Plane Services and one for each line card.

Once the aggregate policies are defined, apply the policies to the Aggregate Control Plane Services by entering the **service-policy** command from the aggregate control-plane configuration mode. This is shown in the following example:

```
router(config)# control-plane
! Apply CoPP QoS policy
router(config-<<???>>)# service-policy input service_policy_name
```

Finally, the distributed CoPP policies can be activated for each line card using the **service-policy** command from the **control-plane slot** configuration mode.

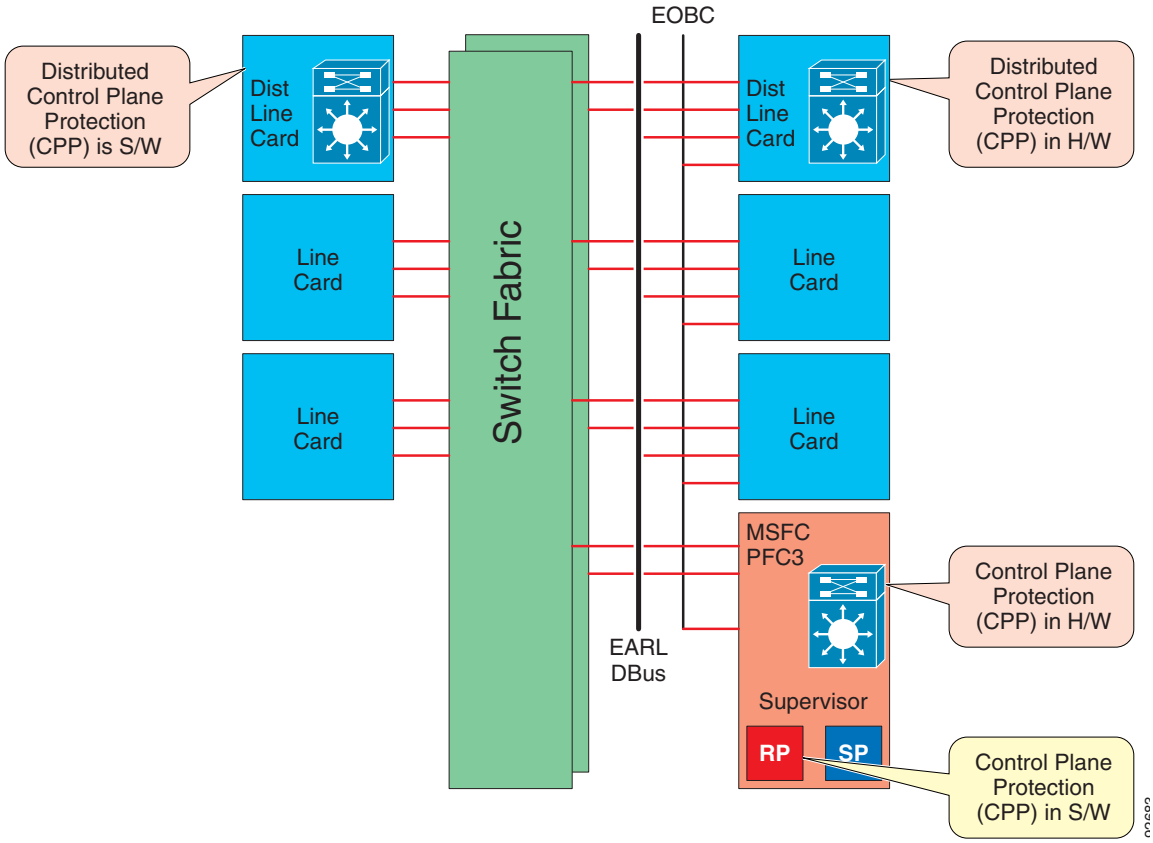
```
router(config)# control-plane slot slot_number
router(config-<<???>>)# service-policy input service_policy_name
```

Supervisor 720 (Catalyst 6500 and Cisco 7600) CoPP

The implementation of CoPP on Supervisor 720 (Catalyst 6500 Series Switches and the Cisco 7600 Series Routers) differs from Distributed CoPP or Centralized CoPP, primarily because it is configured like centralized CoPP but the actual implementation is distributed.

From a configuration point of view, once the CoPP policies are defined they are activated under the **control-plane** configuration mode, similar to centralized CoPP implementations. But from an implementation perspective, the CoPP polices are applied at the RP level, and they get automatically pushed to the Policy Feature Card (PFC) and each Distributed Forwarding Card (DFC). This CoPP model is illustrated in Figure 5.

Figure 5 Centralized CoPP with Distributed Implementation



CoPP at the RP is performed in software while on the PFC and DFCs it is processed in hardware, with no performance degradation or increased latency. In this way CoPP on Sup720 provides two layers of protection. First, at wire speed on the PFC and DFCs, and second, at the RP level. This helps to ensure that only the amount of traffic specified by the user actually reaches the control plane.

It is important to note that, even though this is a distributed model, on Sup720 the same policy is applied on both levels. This is in contrast to distributed CoPP, which allows the definition of different CoPP policies for each line card.

Another important characteristic of CoPP on Supervisor 720 is that police rates can be expressed as bps (bits per second), pps (packets per second), or as a percentage of link utilization. For the exact syntax and options for the Supervisor 720 **police** command, see the following website:

http://www.cisco.com/en/US/products/hw/switches/ps708/products_command_reference_chapter09186a00801eaeca.html#wp1083056

For more information about CoPP and other DoS protection controls available on the Supervisor 720, see the following website:

http://www.cisco.com/en/US/partner/products/hw/switches/ps708/products_configuration_guide_chapter09186a0080435872.html

Expected Effectiveness

CoPP effectively shields the control plane from direct attacks and distributed DoS attacks by blocking undesirable traffic and by rate limiting traffic from authorized protocols and sources. Unlike rACLs, CoPP can even mitigate attacks that use packets based on authorized protocols and sources and that may fall into a permitted traffic class. In most cases, its rate limiting capability can effectively block most of the packets belonging to such attacks. However, because it is not always possible to precisely define good and bad traffic, there are circumstances in which some legitimate packets may be discarded as well.

In the case of non-distributed CoPP, an attack coming from a single interface may take all the resources available for a particular traffic class, which may result in dropping legitimate packets arriving from other ports. Distributed CoPP helps in this situation by limiting the number of control plane packets forwarded from one line card to the central switch engine. In this way, the effects of an attack coming from a single interface are contained within the same line card, leaving the ports of the other line cards unaffected. However, other ports within the same line card can still be negatively affected.

There are other circumstances in which legitimate packets may be discarded with Distributed CoPP. Depending on the distributed control plane policy configuration, the combined rates of all line cards may be over-subscribed compared to the aggregate control plane rates. In that case, the total amount of control plane packets received from all the line cards on a router may exceed aggregate control plane levels As a result, legitimate packets may be discarded.

CoPP Deployment Recommendations

By protecting the RP, CoPP helps ensure router and ultimately network stability during an attack. For this reason, CoPP should be deployed on all routers as a key protection mechanism.

Because CoPP filters traffic, it is critical to gain an adequate level of understanding about the legitimate traffic destined to the RP prior to deployment. CoPP policies built without proper understanding of the protocols, devices or required traffic rates involved may block critical traffic. This has the potential of creating a denial of service (DoS) condition. Determining the exact traffic profile needed to build the CoPP policies might be difficult in some networks. For this reason, this document describes a conservative methodology for deploying CoPP using iterative ACL configurations to help identify and to incrementally filter traffic.

Developing a CoPP Policy

Before developing an actual CoPP policy, required traffic must be identified and separated into different classes. Multiple classification schemes can be used, but one recommended methodology involves classifying traffic into distinct groups based on relative importance. This approach was previously illustrated with a simple classification example composed of four basic classes: critical, normal, undesirable, and default.

This section uses a more advanced classification criteria that groups traffic based on relative importance and traffic type. This example uses ten different classes, which provide greater granularity, and is more suitable for real world environments. It is important to note that although you can use this example as a reference, the actual number and type of classes needed for a given network may differ and should be selected based on local requirements, security policies, and a thorough analysis of baseline traffic.

This example defines the following ten traffic classes:

1. Border Gateway Protocol (BGP)

   This class defines traffic that is crucial to maintaining neighbor relationships for BGP routing protocol, such as BGP keepalives and routing updates. Maintaining BGP routing protocol is crucial to maintaining connectivity within a network or to an ISP. Sites that are not running BGP do not use this class.

2. Interior Gateway Protocol (IGP)

   This class defines traffic that is crucial to maintaining IGP routing protocols such as 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. Interactive Management

   This class defines interactive traffic that is required for day-to-day network operations. This class includes light volume traffic used for remote network access and management. For example, Telnet, Secure Shell (SSH), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), and Terminal Access Controller Access Control System (TACACS).

4. File Management

   This class defines high volume traffic used for software image and configuration maintenance. This class includes traffic generated for remote file transfer. For example, Trivial File Transfer Protocol (TFTP) and File Transfer Protocol (FTP).

5. Reporting

   This class defines traffic used for generating network performance statistics for reporting. This class includes traffic required for using Service Assurance Agent (SAA) to generate ICMP with various DSCP settings to report on response times within various QoS data classes.

6. Monitoring

   This class defines traffic used for monitoring a router. This kind of traffic should be permitted but should never be allowed to pose a risk to the router. With CoPP, this traffic can be permitted but limited to a low rate. Examples include packets generated by ICMP echo requests (**ping**) and the **traceroute** command.

7. Critical Applications

   This class defines application traffic that is crucial to a specific network. The protocols that might be included in this class include generic routing encapsulation (GRE), Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), Session Initiation Protocol (SIP), Data Link Switching (DLSw), Dynamic Host Configuration Protocol (DHCP), IPsec, and multicast traffic.

8. Layer 2 Protocols

   This class defines traffic used for Address Resolution Protocol (ARP). Excessive ARP packets can potentially monopolize RP resources, starving other important processes. CoPP 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** command.

9. Undesirable

   This explicitly identifies unwanted or malicious traffic that should be dropped and denied access to the RP. For example, this class can contain packets from a well-known worm. This class is particularly useful when specific traffic destined to the router should always be denied rather than be placed into a default category. Explicitly denying traffic allows you to collect rough statistics on this traffic using **show** commands and thereby offers some insight into the rate of denied traffic.

10. Default

    This class defines all remaining traffic destined to the RP that does not match any other class. MQC provides the Default class so that you can specify how to treat traffic that is not explicitly associated with any other user-defined classes. It is desirable to give such traffic access to the RP but 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. After this traffic is identified, further analysis can be performed to classify it. If needed, the other CoPP policy entries can be updated to account for this traffic.

Deployment Methodology

To implement the conservative methodology recommended for deploying CoPP, complete the following steps:

Step 1 Determine the classification scheme for your network.

Identify the known protocols that access the RP and divide them into categories using the most useful criteria for your specific network. For example, the nine categories given in the example in this section (BGP, IGP, Management, Reporting, Critical Applications, Layer 2 Protocols, Undesirable, and Default) use a combination of relative importance and traffic type. Select a scheme suited to your specific network, which may require a larger or smaller number of classes.

Step 2 Define classification ACLs.

Develop an ACL for each class identified in Step 1, including one for the Default class.

Configure each ACL to permit all known protocols in its class that require access to the RP. At this point, each ACL entry should have both source and destination addresses set to **any**. In addition, the ACL for the Default class should be configured with a single entry: **permit ip any any**. This will match traffic not explicitly permitted by entries in the other ACLs.

Once the ACLs have been configured, create a **class-map** for each class defined in Step 1, including one for the Default class. Then assign each ACL to its corresponding **class-map**.

Note In this step you should create a separate **class-map** for the Default class, 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 you to identify traffic not yet classified as part of another class.

Each class map should then be associated with 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 the identified traffic and adjust the classification.

Ideally, the classification performed in Step 1 identified all required traffic destined to the router. However, realistically, not all required traffic will be identified prior to deployment and the **permit ip any any** entry in the Default class ACL 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 see the entries in the ACLs that are in use, and to identify any additional traffic sent to the RP. To analyze the unclassified traffic you can use one of the following techniques:

- General ACL classification as described in *Characterizing and Tracing Packet Floods Using Cisco Routers*, available at the following website:

  http://www.cisco.com/warp/public/707/22.html#topic2

- Packet analyzers
- rACLs

Once traffic has been properly identified, adjust the class configuration accordingly. Remove the ACL entries for those protocols that are not used. Add a **permit any 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 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 172.68.0.0/16 where applicable.

This step provides data points for devices or users from outside the CIDR block that might be accessing the equipment. External BGP (eBGP) peer will require an exception because the permitted source addresses for the 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 authorized source addresses.

Increasingly limit the source address in the classification ACLs to only permit sources that communicate with the RP. For instance, only known network management stations should be permitted to access the SNMP ports on a router.

Step 6 Refine CoPP 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.

At this point, you may decide to remove the **class-map** and ACL used for the Default class. If so, you should also replace the previously defined policy for the Default class by the **class-default** policy.

Design Considerations and Limitations

This section describes some design considerations and limitations to keep in mind when deploying CoPP and includes the following topics:

Class Matching

When designing CoPP policies it is critical to understand how packets will be classified into each traffic class. The configuration order of classes within a policy, or the number of match statements in a class, can affect the way packets are classified.

The first thing to consider is that classes in a policy are inspected sequentially. Packets are compared against the first configured class within the policy. If there is a match, the associated control plane policy is applied. Otherwise, the remaining classes are sequentially inspected until a match occurs or until the end of the policy is reached. For this reason, and to avoid packets being classified into incorrect classes, classes within a policy should be configured from more specific to more general.

Another important aspect to consider is that a single class may contain more than one match criteria. By default, a packet must meet all the specified match criteria (match-all) in order to be considered a member of a class. However some traffic classes may only require packets to meet one of the specified match criteria (match-any). In general, traffic destined to the undesirable class should use the **match any** command in class configuration mode because you want to drop all traffic that has any of the match criteria specified in this class.

Fragmented Packets

Fragmented packets can be used by an attacker to avoid detection by protection mechanisms, to attempt to bypass existing ACLs, and to increase the effects of an attack by consuming more resources through fragment reassembly.

ACLs have a **fragments** keyword that enables specialized handling behavior for fragmented packets. In general, non-initial fragments that match the L3 statements (regardless of the L4 information) in an ACL are affected by the **permit** or **deny** statement of the matched entry. However using the **fragments** keyword can force ACLs to either deny or permit non-initial fragments with more granularity.

In the CoPP context, filtering fragments adds an additional layer of protection against a DoS attack that uses non-initial fragments (for example, FO > 0). However, note that a CoPP deny policy for fragments will deny all subsequent (or non-initial) fragments from accessing the router.

The following ACL filters all non-initial fragments for each of the protocols listed, and provides basic classification if a fragmentation attack occurs because the access list entries log packet counts for each matching packet.

```
access-list 101 deny tcp any any fragments
access-list 101 deny udp any any fragments
access-list 101 deny icmp any any fragments
```

The most effective way to apply this ACL is to create a new class, which might be called coppclass-fragments. The initial policy associated with this class can be a drop policy and if needed changed to a rate limited policy.

As explained earlier, the order of defining classes within a policy is important. Therefore, the fragments class must be defined as the first 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.

The following is a sample configuration listing for the required configuration for the coppclass-fragments class:

```
class-map coppclass-fragments
     match access-group 110
class-map coppclass-bgp
     match access-group 120
…….
policy-map copp-policy
     class coppclass-fragments
     police 8000 conform-action drop exceed-action drop
     !if the unconditional packet drop command is supported, you can configure drop
     class coppclass-bgp
     ! no operation needs to be specified as this class has unrestricted access to the RP
…….
```

Under certain circumstances, a valid session might require fragmentation and therefore be filtered if fragments are denied through a CoPP drop policy. For example, routers acting as IPSec termination devices will often process fragmented packets. In this case, rate limiting, rather than dropping, should be used. This is shown in the following example:

```
policy-map copp-policy
     class coppclass-fragments
     police 8000 conform-action transmit exceed-action drop
```

Type of Service

Associating a packet with an IP Precedence or DSCP marking allows users to classify traffic based on these values.

- IP DSCP value: first 6 bits in the 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 bare minimum, packets queued for RP 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 CoPP policy can be developed to help protect the router from invalid TOS values. For example, receive path traffic from unknown sources with high TOS values (6 or 7, which are usually reserved for routing protocols), can safely be dropped. True routing protocol traffic will pass unaffected and therefore be properly classified.

The TOS assurance policy can take on several forms. However, the simplest and most effective way to drop high precedence traffic from unexpected sources is to define a new traffic classification.

For example, the ToSDrop class can define traffic based on precedence, rather than by ACL. Traffic will be dropped if precedence matches 6 or 7. Valid TOS 6/7 traffic received from routing peers will be identified and handled by either the BGP or IGP class. For this reason, the ToSDrop class must be defined within the CoPP policy after the critical routing protocols that use TOS 6, TOS 7 and other classes that rely on TOS 6 and TOS 7 traffic are defined. The following configuration listing shows an example of the configuration required:

```
class-map match-all coppclass-bgp
     match access-group 101
class-map match-all coppclass-igp
     match access-group 102
class-map match-any TOSDrop
     match ip precedence 6 7
policy-map cpp
     class coppclass-bgp
     class coppclass-igp
     class ToSDrop
     police 8000 1500 1500 conform-action drop exceed-action drop
```

Another option is to define ToS 6 or 7 as matching criteria for the Undesirable class. This implies that all traffic received from other classes is trusted and that there is no concern that high precedence attack traffic will fall into another class. The following listing illustrates the configuration required:

```
class-map match-any coppclass-undesirable
     match access-group 123
     match ip precedence 6 7
```

Policing ARP Traffic

All ARP traffic is sent to the RP, and a sudden surge in ARP packets can result in high RP CPU utilization. Excessive ARP traffic can result from an ARP storm used in a DoS attacks or from a high volume of legitimate ARP requests generated from end stations. Service Providers often experience a sudden surge of incoming ARP packets destined to their edge routers from one of their customer sites during a worm outbreak. ARP packets can potentially monopolize RP resources, starving other important processes.

Protocol-specific classification and rate limiting can be applied to ARP packets. Following the methodology described in this document, ARP packets can be placed into the Layer-2 Protocols class, as shown in the following example configuration:

```
class-map match-any coppclass-layer2
     match access-group 121
     match protocol arp
```

Once properly classified, appropriate policing can be configured, as in the following example:

```
class-map coppclass-layer2
     police 8000 1500 1500 conform-action transmit exceed-action drop
     policy-map copp-policy
```

This rate limits the ARP traffic so it is dropped if it exceeds 8 Kbps.

Note ARP policing on Supervisor 720 (Catalyst 6500 and Cisco 7600) is supported through a specific rate limiter and not through CoPP. For more information about the rate limiters available on the Supervisor 720, see the following website: http://www.cisco.com/en/US/partner/products/hw/switches/ps708/products_configuration_guide_chapter09186a0080435872.html

Modular QoS Restrictions

CoPP depends on MQC for configuring packet classification and policing. Therefore, restrictions that apply to MQC also apply to CoPP. Also, only two MQC actions are supported in policy maps: **police** and **drop**.

Features that require network-based application recognition (NBAR) classification may not work well at the control plane level.

Only the following classification (match) criteria are supported:

- Standard and extended IP access lists
- **match ip dscp** command
- **match ip precedence** command
- **match protocol arp** command

Considerations on Supervisor 720 (Catalyst 6500 and Cisco 7600)

As described in the "Supervisor 720 (Catalyst 6500 and Cisco 7600) CoPP" section on page 26, the implementation of CoPP on Supervisor 720 differs from the models implemented on other platforms. Primarily, this is because it is configured as centralized CoPP while the actual implementation is

distributed. With Supervisor 720, CoPP policies are configured in the same way as non-distributed CoPP. However, the policies get applied at the RP level and automatically pushed into the PFC and into every DFC present in the system. As a result, the same CoPP policy gets applied at two distinct levels.

There are also other important differences that need to be noted:

Because CoPP relies on the QoS implementation, CoPP policies are downloaded to the PFC and DFCs only if QoS is enabled. For this reason, make sure the command **mls qos** is enabled at the global configuration mode for the PFC and each DFC where CoPP is required.

On Sup720, ARP policing is done with a QoS rate limiter rather than CoPP. The match criteria available for traffic classification includes only the following:

- **match access- group** (standard and extended ACLs)
- **match dscp**
- **match ip precedence**

There is no **match protocol arp** for CoPP on Sup720. ARP policing should be configured with a QoS rate limiter using the **mls qos protocol arp police** *bps* command.

Prior to Cisco IOS software Release 12.2(18)SXE, only one **match** criteria was allowed for each traffic class. When using one of these earlier releases, to define multiple match rules with a **match-any** criteria, split the match **access-group** statements among multiple class maps instead of grouping them together.

Prior to Cisco IOS software Release 12.2(18)SXE, the MQC **class-default** was not supported on Sup720. This is a minor limitation because the **class-default** could be emulated with a normal class configured with an **ip permit any** rule.

Omitting the policy parameters in a class causes the class to be handled by software-based CoPP. Use the **police** command and set the policy parameters to ensure the class is handled by hardware-based CoPP.

Currently, multicast packets are only handled by the software-based CoPP at the RP level. However, there are CPU rate limiters available that can rate limit multicast packets to the CPU in hardware. These CPU rate limiters include the Multicast FIB-miss rate limiter and the Multicast Partial-SC rate limiter.

With PFC3A, egress QoS and CoPP cannot be configured at the same time. In this situation, CoPP is performed in software, and a warning message is generated.

In the rare situation where a large QoS configuration is being used, it is possible that the system may run out of TCAM space. When this scenario occurs, CoPP may be performed in software.

Supervisor 720 allows the use of the following match criteria for packet classification:

- access-group (IPv4 and IPv6 Access Control Lists)
- protocol ip
- protocol ipv6
- ip dscp
- ip precedence
- dscp
- precedence

Understanding Control and Management Plane Traffic

Care must be taken to ensure that CoPP policies do not filter critical traffic such as routing protocols or traffic required for management access to the routers. Filtering required traffic could disable remote access to the router, and connectivity would have to established with a local console connection.

It is recommended that this feature be tested before deployment in a lab configuration that mirrors the actual deployment as closely as possible.

Platform and Software Availability

CoPP has been introduced in 12.2(18)S for Cisco 7200, Cisco 7300, Cisco 7500; integrated in 12.3(4)T for Cisco 1710, Cisco 1721, Cisco 2610XM-2611XM, Cisco 2620XM-2621XM, Cisco 2650XM-2651XM, Cisco 2691, Cisco 3725, Cisco 3745, Cisco 7200, and Cisco 7301; integrated in 12.0(29)S for Cisco 7500 and Cisco 7200; integrated in 12.3(11)XL for Cisco 2610XM-2611XM, Cisco 2620XM-2621XM, Cisco 2650XM-2651XM, Cisco 2691, Cisco 2811, Cisco 2821, Cisco 2851, Cisco 3725, Cisco 3745, Cisco 3825, Cisco 3845; integrated in 12.2(18)SXD1 for Supervisor Engine 720 (Catalyst 6500 and Cisco 7600).

Support for Cisco 1812 has been introduced in 12.3(8)YH, and for Cisco 1841 in 12.3(8)YG.

Support for ONS15530 has been introduced in 12.2(18)SV. Support for both ONS15530 and ONS15540 has been introduced in 12.2(22)S.

Support for distributed control plane services on the Cisco 12000 series Internet router was added in 12.0(30)S.

Aggregate policing is supported in Cisco IOS software Release 12.0(29)S, Cisco IOS software Release 12.2(18)S, and Cisco IOS software Release 12.3(4)T and later releases.

Distributed policing is supported only in Cisco IOS software Release 12.0(30)S and later Cisco IOS software 12.0S Releases.

Dependencies and Prerequisites

The following are the dependencies and prerequisites required for implementing CoPP:

- Understanding the concepts and general configuration procedure (class map and policy map) for applying quality-of-service (QoS) policies on a router.
- Enabling QoS on Supervisor 720.

  CoPP policies are downloaded to the PFC and DFCs only if QoS is enabled. For this reason, make sure the **mls qos** command is enabled at the global configuration mode.

CoPP Sample Configuration

The following example shows how to develop a CoPP policy and how to apply it to protect the control plane of a router.

In this example, the control plane traffic is classified based on relative importance and traffic type. Nine classes are defined, each of which is associated with a separate extended ACL:

- BGP (coppacl-bgp)—BGP traffic
- IGP (coppacl-igp)—OSPF traffic
- Interactive Management (coppacl-interactivemanagement)—Remote access and management traffic such as TACACS, SSH, SNMP, and NTP.
- File Management (coppacl-filemanagement)—Remote file transfer traffic such as TFTP and FTP.
- Reporting (coppacl-reporting)—SAA generated ICMP requests from SAA source routers

- Monitoring (coppacl-monitoring)—ICMP and traceroute traffic
- Critical Applications (coppacl-critical-app)—HSRP and DHCP traffic
- Undesirable Traffic (coppacl-undesirable)—Explicitly denies unwanted traffic (for example, Slammer worm packets)
- Default (no ACL needed)—All traffic received by the control plane that has not been otherwise identified.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Sample basic ACLs for CoPP classification
!
! In this example, BGP is used and must be classified
ip access-list extended coppacl-bgp
 remark BGP traffic class
! allow BGP from a known peer to this router's BGP TCP port
 permit tcp host 192.168.1.1 host 10.1.1.1 eq bgp
! allow BGP from a peer's BGP port to this router
 permit tcp host 192.168.1.1 eq bgp host 10.1.1.1
!
! For your IGP class, OSPF is the IGP used in this example
ip access-list extended coppacl-igp
 remark 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 Interactive Management class is for traffic that is required for accessing
! and managing the router, in this example, TACACS, ssh, telnet,
! snmp, and ntp is classified in this class
ip access-list extended coppacl-interactivemanagement
 remark CoPP interactive management traffic class
! permit return traffic from TACACS host
 permit tcp host 10.2.1.1 host 10.1.1.1 established
! ssh access to the router from a subnet
 permit tcp 10.2.1.0 0.0.0.255 host 10.1.1.1 eq 22
! SNMP access from the NMS host to the router
 permit udp host 10.2.2.2 host 10.1.1.1 eq snmp
! Allow the router to receive NTP packets from a known clock source
 permit udp host 10.2.2.3 host 10.1.1.1 eq ntp
!
! The File Management class is for file transfer traffic required for software
! and configuration maintenance, in this example, TFTP and FTP is classified in this class
ip access-list extended coppacl-filemanagement
 remark CoPP file management traffic class
! Allow router initiated FTP (active and passive)
 permit tcp 10.2.1.0 0.0.0.255 eq 21 host 10.1.1.1 gt 1023 established
 permit tcp 10.2.1.0 0.0.0.255 eq 20 host 10.1.1.1 gt 1023
 permit tcp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt 1023 established
! Allow router initiated TFTP
 permit udp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt 1023
!
! 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 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 routers
 permit icmp host 10.2.2.4 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
! particular 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 10.2.2.8 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
```

After the control plane traffic has been classified, the next step is to define the policy action for each traffic class. In this example, the intention is to deploy a policy that protects the router while limiting the risk of dropping critical traffic. To that end, CoPP policies are configured to permit each traffic class with an appropriate rate limit. Table 1 shows the parameters used in the CoPP policies.

Table 1 Sample CoPP Policy

Traffic class	Rate (bps)	Conform action	Exceed action
BGP	80,000	Transmit	Drop
IGP	N/A	Transmit	Transmit
Interactive Management	10,000,000	Transmit	Drop
File Management	N/A	Transmit	Transmit
Reporting	500,000	Transmit	Drop
Monitoring	500,000	Transmit	Drop
Critical Apps	500,000	Transmit	Drop
Undesirable	50,000	Drop	Drop
Default	10,000,000	Transmit	Drop

Note The rates defined in Table 1 were successfully tested on a Cisco 7200 VXR Series Router with NPE-G1. Note that the values here presented are solely for illustration purposes; every environment has different baselines.

The following is the policy for the configuration described in Table 1:

```
! Define a class for each "type" of traffic and associate it with an 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-interactivemanagement
  match access-group name coppacl-interactivemanagement
class-map match-all coppclass-filemanagement
  match access-group name coppacl-filemanagement
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-undesirable
  match access-group name coppacl-undesirable
!
! This is the actual policy. Depending on class of traffic, rates and associated actions
! are defined
policy-map copp-policy
!
! BGP traffic is limited to a rate of 80,000 bps, if traffic exceeds
! that rate it is dropped. NOTE: In this example BGP traffic is rate-limited
! to control attacks based on BGP packets. Once the normal rates are determined,
! and depending on the hardware platform used, it's recommended you consider
! readjusting the rate-limiting parameters.
  class coppclass-bgp
     police 80000 8000 8000 conform-action transmit exceed-action drop
!
! 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 your IGP traffic, you may
! consider setting a rate-limit to further protect your router.
  class coppclass-igp
!
! Interactive Management traffic is limited to a rate of 10,000,000 bps,
! if traffic exceeds that rate it is dropped.
  class coppclass-interactivemanagement
     police 10000000 100000 100000 conform-action transmit exceed-action drop
!
! File Management traffic will not be limited in this example either therefore no
! operation needs to be specified in this class. NOTE: As with the IGP
! class, once normal rates are determined for your file management traffic,
! you may consider setting a rate-limit to further protect your router.
  class coppclass-filemanagement
!
! Reporting traffic is limited to a rate of 500,000 bps, if traffic exceeds
! that rate it is dropped
  class coppclass-reporting
     police 500000 5000 5000 conform-action transmit exceed-action drop
!
! Monitoring traffic is limited to a rate of 500,000 bps, if traffic exceeds
```

```
! that rate it is dropped
  class coppclass-monitoring
     police 500000 5000 5000 conform-action transmit exceed-action drop
!
! critical-app traffic is limited to a rate of 500,000 bps, if traffic
! exceeds that rate it is dropped
  class coppclass-critical-app
     police 500000 5000 5000 conform-action transmit exceed-action drop
!
! This policy drops all traffic categorized as undesirable, regardless
! of rate.
  class coppclass-undesirable
     police 50000 1500 1500 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 10,000,000 bps and violations of that limit
! are dropped.
  class class-default
     police 10000000 100000 100000 conform-action transmit exceed-action drop
….
! Applies the defined CoPP policy to the control plane
Router(config)# control-plane
Router(config-cp)# service-policy input copp-policy
```

Useful Debugs and Show Commands

The **show access-lists** command displays a counter for each ACL entry (ACE), which increments when traffic matches a particular entry. This information can be used to develop a CoPP policy that ensures that identified traffic is matching as expected.

The following is sample output from the **show access-lists** command:

```
router# show access-lists
Extended IP access list coppacl-bgp
    10 permit tcp host 192.168.1.1 host 10.1.1.1 eq bgp (4 matches)
    20 permit tcp host 192.168.1.1 eq bgp host 10.1.1.1 (1 match)
Extended IP access list coppacl-critical-app
    10 permit ip any host 224.0.0.2
    20 permit udp host 0.0.0.0 host 255.255.255.255 eq bootps
    30 permit udp host 10.2.2.8 eq bootps any eq bootps
Extended IP access list coppacl-filemanagement
    10 permit tcp 10.2.1.0 0.0.0.255 eq ftp host 10.1.1.1 gt 1023 established
    20 permit tcp 10.2.1.0 0.0.0.255 eq ftp-data host 10.1.1.1 gt 1023
    30 permit tcp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt 1023 established
    40 permit udp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt 1023
Extended IP access list coppacl-igp
    10 permit ospf any host 224.0.0.5 (16 matches)
    20 permit ospf any host 224.0.0.6
    30 permit ospf any any (24 matches)
Extended IP access list coppacl-interactivemanagement
    10 permit tcp host 10.2.1.1 host 10.1.1.1 established
    20 permit tcp 10.2.1.0 0.0.0.255 host 10.1.1.1 eq 22
    30 permit udp host 10.2.2.2 host 10.1.1.1 eq snmp
    40 permit udp host 10.2.2.3 host 10.1.1.1 eq ntp
Extended IP access list coppacl-monitoring
    10 permit icmp any any ttl-exceeded
    20 permit icmp any any port-unreachable
    30 permit icmp any any echo-reply
    40 permit icmp any any echo
```

```
Extended IP access list coppacl-reporting
    10 permit icmp host 10.2.2.4 host 10.1.1.1 echo
Extended IP access list coppacl-undesirable
    10 permit udp any any eq 1434
```

The **show policy-map control-plane** command is a valuable tool for developing site-specific policies, monitoring statistics for the control plane policy, and troubleshooting CoPP. 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 is sample output from the **show policy-map control-plane** command:

```
router# show policy-map control-plane

Service-policy input: cpp

Class-map: coppclass-bgp (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: access-group name coppacl-bgp
  police:
      cir 80000 bps, bc 8000 bytes
    conformed 0 packets, 0 bytes; actions:
      transmit
    exceeded 0 packets, 0 bytes; actions:
      drop
    conformed 0 bps, exceed 0 bps

Class-map: coppclass-igp (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps
  Match: access-group name coppacl-igp

…
Class-map: class-default (match-any)
  744 packets, 53218 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any
  police:
      cir 10000000 bps, bc 100000 bytes
    conformed 764 packets, 54831 bytes; actions:
      transmit
    exceeded 0 packets, 0 bytes; actions:
      drop
    conformed 1000 bps, exceed 0 bps
```

Management and Telemetry

The CBQoSMIB, the primary accounting mechanism for MQC-based policies, provides SNMP MIB support for monitoring and managing CoPP. This functionality is available from Cisco IOS Software Releases 12.2(5)S and 12.3(7)T.

For additional information about CBQoSMIB, see the following website:

ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CLASS-BASED-QOS-MIB-CAPABILITY.my

For additional information about the CBQoSMIB configuration and statistics, see the following website:

ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CLASS-BASED-QOS-MIB.my

Related Tools

Infrastructure Protection ACLs (iACLs) are Access Control Lists (ACLs) that complement CoPP. iACLs are filters deployed at the edge routers to prevent and mitigate direct infrastructure attacks by explicitly permitting only authorized traffic to the infrastructure equipment, while allowing transit traffic.

Receive ACLs (rACL), which are similar to CoPP, implement a filter mechanism that protect the RP on high end routers from unnecessary traffic destined to the RP itself that could potentially affect system performance. However, rACLs can only bock or permit traffic, and cannot rate limit authorized traffic as CoPP does. In addition, the distributed CoPP model provides greater flexibility and efficiency, and it is available on some platforms. For these reasons, CoPP is preferable to rACLs, which should only be deployed on platforms that do not support CoPP.

More Information

For more information about Control Plane Policing, see the following website:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1838/products_feature_guide09186a00801afad4.html

For more information about deploying Control Plane Policing, see the following website:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1838/products_white_paper09186a0080211f39.shtml

Control Plane Protection

Control Plane Protection is a security feature that extends the policing functionality provided by the software-based Control Plane Policing (CoPP) feature. The CoPP feature controls the rate at which control plane traffic is sent to the route processor in Cisco IOS software-based devices. Control Plane Protection extends this policing functionality by dividing the control plane into three control plane sub-interfaces and allowing the enforcement of separate rate-limiting policies. In addition, Control Plane Protection incorporates port-filtering and queue-thresholding. Port-filtering is a mechanism for the early dropping of packets that are directed to closed or non-listened IOS TCP/UDP ports. Queue-thresholding is a mechanism that limits the number of packets per protocol hold in the control-plane input queue, preventing the input queue from being overwhelmed by any single protocol traffic. This section describes how Control Plane Protection is implemented, and includes the following topics:

- Control Plane Protection Technology Overview
- Threat Vectors
- Functional Overview
- Expected Effectiveness
- Control Plane Protection Deployment Recommendations
- Design Considerations and Limitations
- Platform and Software Availability
- Dependencies and Prerequisites
- Control Plane Protection Sample Configuration
- Useful Debugs and Show Commands

Control Plane Protection Technology Overview

The Control Plane Protection is a feature that extends the policing functionality of the software-based version of Control Plane Policing (CoPP) by providing an additional layer of protection to the control plane. With Control Plane Protection, the first layer of protection is provided by CoPP at an aggregate level, which controls all packets destined to the control plane. After traffic is processed by CoPP, it is then handed to Control Plane Protection, the second layer of protection, which divides the traffic into three categories. Each category is processed by a control plane sub-interface with independent rate-limiting policies. This dual layer of protection provides a control hierarchy that allows for finer policy definition and enforcement.

Note Control Plane Protection is available only on platforms that support software-based CoPP. This feature is currently not available on platforms with hardware-based or distributed CoPP.

The three control plane sub-interfaces implemented by Control Plane Protection are the following:

- Control-plane host subinterface—Handles all control plane IP packets that are destined to any of the IP addresses configured on the router interfaces. Examples of traffic falling in this category include tunnel termination traffic, management traffic or routing protocols such as SSH, SNMP, BGP, OSPF, and EIGRP. All host traffic terminates on and is processed by the router.
- Control-plane transit subinterface—Receives all IP packets that are software switched by the route processor. This means packets that are not directly destined to the router itself but rather traffic traversing through the router that require process switching.
- Control-plane CEF-exception subinterface—Receives all IP packets that are either redirected as a result of a configured input feature in the CEF packet forwarding path for process switching or directly queued in the control plane input queue by the interface driver (that is, ARP, L2 keepalives, and all non-IP host traffic).

In addition, Control Plane Protection enhances the protection of the control plane host subinterface by implementing Port-filtering and Queue-thresholding. Port-filtering is a feature that can be applied only to the control plane host subinterface, which automatically drops packets directed toward closed or non-listened UDP/TCP ports on the router. Queue-thresholding is another feature that can be applied only to the control plane host subinterface, and that limits the number of unprocessed packets per protocol, preventing the input queue from being overwhelmed by any single protocol traffic.

Threat Vectors

As CoPP, Control Plane Protection helps protect the RP of Cisco IOS software-based routers by filtering unwanted traffic and by rate-limiting the traffic expected by the control plane. This shields the control plane from traffic that might be part of DoS or other attacks, helping maintain network stability even during attack conditions.

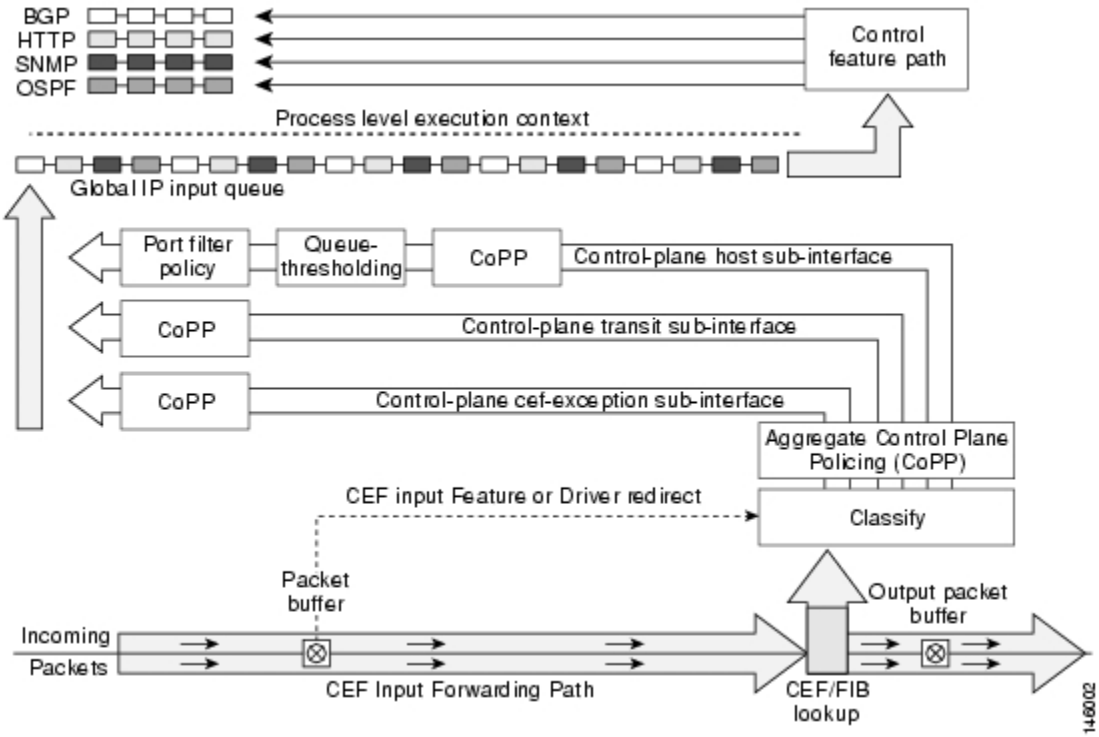
In addition, Control Plane Protection ability to divide the control plane traffic and rate-limit each traffic type individually, gives you greater traffic control for attack mitigation. Port-filtering and Queue-thresholding also provide for a more advanced attack protection. On one hand, Port-filtering

shields the RP from packets directed to closed or non-listened TCP/UDP ports, mitigating attacks attempting to spoof legitimate traffic permitted by CoPP. On the other hand, Queue-thresholding limits protocol queue usage mitigating attacks designed to overwhelm the input queue with the flooding of a single protocol.

Functional Overview

The Control Plane Protection feature extends the functionality of CoPP on platforms that only support the software version of this feature. As illustrated in Figure 6, CoPP, Control Plane Protection provides a second layer of protection on top of CoPP. CoPP provides the first layer of protection at an aggregate level, controlling all packets sent to the control plane. After traffic is processed by Aggregate CoPP, packets are handled to the Control Plane Protection module. Control Plane Protection includes a traffic classifier, which intercepts the traffic coming from aggregate CoPP and classifies it into the three control plane subinterfaces (host, transit, CEF-exception). Each control-plane subinterface can have an individual traffic policy in place for finer granularity for traffic policing. In addition, traffic sent to the control plane host subinterface can be further controlled by the Port-filtering and Queue-thresholding policies.

Figure 6 Control Plane Protection



At a very high level, the sequence of events with Control Plane Protection is as follows:

1. A packet enters the router configured with CoPP on an ingress interface.
2. The interface performs the basic input port and QoS services.
3. The packet gets forwarded to the router processor.

4. The router processor makes a routing decision, determining whether or not the packet is destined to the control plane.
5. Packets destined for the control plane are processed by Aggregate CoPP, and are dropped or forward to the Control Plane Path according to the polices for each traffic class. Packets that have other destinations are forwarded normally.
6. Packets sent to the Control Plane Path are intercepted by the Control Plane Protection traffic classifier, which classifies the packets into the corresponding control plane subinterfaces.
7. Packets received by each control plane subinterface are dropped or forward to the Control Plane global input queue according to the configured policies.
8. In addition, packets sent to the control plane host subinterface can be dropped or forwarded according to the Port-filter and Queue-thresholding policies before they are sent to the global input queue.

As mentioned previously, Port-filtering enhances control plane protection by providing for early dropping of packets directed toward closed or nonlistened IOS TCP/UDP ports on the router. To that end, the port-filter maintains a global database of all open TCP and UDP ports on the router, including random ephemeral ports created by applications. The port database is dynamically populated with entries provided by the registered applications as they start listening on their advertised ports either by configuration of an application (that is SNMP) or initiation of an application (that is, TFTP transfer). An MQC class-map using the list of open ports can be configured and a simple drop policy can be applied to drop all packets destined to closed or nonlistened ports. Port-filter class-maps also support direct match of any user configured TCP/UDP port numbers.

The Queue-thresholding feature also enhances control plane protection but by providing a mechanism for limiting the number of unprocessed packets a protocol can have at process-level. As Port-filtering, this feature can only be applied to the control-plane host subinterface. The intent of this feature is to prevent the input queue from being overwhelmed by any single protocol traffic. Per-protocol thresholding follows a protocol charge model. The queue usage of each protocol is limited such that no single misbehaving protocol process can jam the interface hold queue. In this release, only a subset of TCP/UDP protocols can be configured for thresholding. Non-IP and Layer 2 protocols such as ARP and CDP cannot be configured. You can set queue limits for the following protocols:

- BGP—Border Gateway Protocol
- DNS—Domain Name Server lookup
- FTP—File Transfer Protocol
- HTTP—World Wide Web traffic
- IGMP— Internet Group Management Protocol
- SNMP—Simple Network Management Protocol
- SSH—Secure Shell Protocol
- Syslog—Syslog Server
- Telnet—Telnet
- TFTP—Trivial File Transfer Protocol
- Host-protocols—A wild card for all TCP/UDP protocol ports open on the router not specifically matched/configured

In terms of configuration, Control Plane Protection uses the same principles introduced with Control Plane Policing. In fact, enabling Control Plane Protection also requires the configuration of MQC class-maps and policies maps. The only differences are that the CLI for control-plane has been extended

to allow the configuration of individual policies per control-plane subinterface. In addition, the MQC class-map and policy-map CLI was modified to allow for additional types used in port-filter and queue-threshold policies.

As CoPP, to enable a Control Plane Protection policy, first classify the control plane traffic using the **class-map** command. The syntax for this command is as follows:

```
router(config)# class-map [match-any | match-all] traffic_class_name
```

The **class-map** command defines the class map name and enables a configuration mode for defining the class. Use the **match-any** option to specify that one of the match criterion must be met for traffic entering the traffic class to be classified as part of the traffic class. Use the **match-all** option to indicate that all match criterion must be met for traffic entering the traffic class to be classified as part of the traffic class.

Within the traffic class configuration mode, use the **match** command to associate specific traffic with the class. The syntax for this command is as follows:

```
router(config-cmap)# match {access-group | name access-group-name} | protocol arp | ip
precedence | ip dscp
```

After the traffic is classified, you apply a policy action to each class, indicating whether to permit all packets, to drop all packets, or to drop packets crossing a specified rate limit for that particular class. To apply these policy actions use the **policy-map** command, which has the following syntax:

```
router(config)# policy-map service_policy_name
```

The **policy-map** command defines the policy map name and enables a configuration mode for defining the policy. You then use the **class** command to associate one or more traffic classes with the policy. You use the **police** command to define the policy action to apply. The syntax for these commands is as follows:

```
router(config-pmap)# class traffic_class_name
router(config-pmap-c)# police bps-rate [burst-normal] [burst-max] conform-action action
exceed-action action
```

Then, access the control-plane subinterface where you want the policy to be applied by using the **control-plane** command. Finally, apply the Control Plane Protection policy to the intended control-plane subinterface by using the **service-policy** command from **control-plane** configuration mode. The syntax for these commands is as follows:

```
router(config)# control-plane [host|transit|cef-exception]
router(config-cp)# service-policy {input | output} service_policy_name
```

Port-filter Policy

Optionally, a port-filter policy can be applied to the control-plane host subinterface for additional control-plane protection. The port-filter policy helps by dropping packets that are directed toward closed on nonlistened TCP/UDP ports on the router.

To enable a port-filter policy, you must first define a traffic class for the traffic you want to block. This traffic class is created by using the new **class-map type port-filter** command.

```
router(config)# class-map type port-filter [match-all | match-any] class_name
```

The **match-all** option performs a logical AND on the match criteria, while **match-any** performs a logical OR on the match criteria.

After a port-filter class-map is created, you must define the TCP/UDP match criteria for the traffic you want to block. Under the class-map configuration, use the **match** command to define the match criteria.

```
router(config-cmap)# match {closed-ports|not|port} {TCP|UDP} 0-65535
```

Use the **closed-ports** option to automatically match on all closed-ports on the router. The port option allows you to manually specify a TCP/UDP port on which to match. The **TCP** and **UDP** options specify the TCP or UDP port on which to match.

After a port-filter class-map is defined, you must configure a port-filter service policy. This policy associates the port-filter class-map previously defined with a policy action. Drop is the only policy action supported for port-filter service policies. The port-filter service policy is configured with the **policy-map type port-filter** command.

```
router(config)# policy-map type port-filter policy_map_name
```

Then, under the policy map configuration associate the previously defined port-filter class-map by using the **class** command, and apply the drop action on the class.

```
router(config-pcmap)# class class_name
router(config-cmap)# drop
```

Finally, the port-filter service policy can be applied to the control-plane host subinterface. Use the **control-plane host** command to enter the control-plane host subinterface configuration mode, and associate the port-filter policy with the **service-policy** command.

```
router(config)# control-plane host
router(config-cp)# service-policy input policy_map_name
```

Queue-thresholding Policy

Optionally, a queue-thresholding policy can applied to the control plane host subinterface for additional control plane protection. A queue-threshold policy limits the number of packets for a given higher level protocol allowed in the control plane IP input queue.

To enable a queue-thresholding policy, you must first define a traffic class representing the protocols you want to queue limit. This traffic class is created by using the new **class-map type queue-threshold** command.

```
router(config)# class-map type queue-threshold [match-all | match-any] class_name
```

The **match-all** option performs a logical AND on the match criteria, while **match-any** performs a logical *or* on the match criteria.

After a queue-thresholding class-map is created, you must define the match criteria for the protocols you want to limit. Under the class-map configuration use the **match protocols** command to define the match criteria.

```
router(config-cmap)# match protocol
[bgp|dns|ftp|http|igmp|snmp|ssh|syslog|telnet|tftp|host-protocols]
```

Use the **host-protocols** option to match any open TCP/UDP ports on the router.

After a queue-thresholding class-map is defined, you must configure a queue-thresholding service policy. This policy associates the queue-thresholding class-map previously defined with a queue limit. The queue-thresholding service policy is configured with the **policy-map type queue-threshold** command.

```
router(config)# policy-map type queue-threshold policy_map_name
```

Then, under the policy map configuration associate the previously defined queue-thresholding class-map by using the **class** command, and apply a queue limit on the class. Queue limit range is 0 to 255.

```
router(config-pcmap)# class class_name
router(config-cmap)# queue-limit number
```

Finally, the queue-thresholding service policy can be applied to the control-plane host subinterface. Use the **control-plane host** command to enter the control-plane host subinterface configuration mode, and associate the queue-thresholding policy with the **service-policy** command.

```
router(config)# control-plane host
router(config-cp)# service-policy input policy_map_name
```

Expected Effectiveness

As CoPP, Control Plane Protection can effectively shield the control plane from direct attacks and distributed DoS attacks by blocking undesirable traffic and by rate limiting the traffic expected by the control plane. Control Plane Protection can also mitigate attacks that use packets based on authorized protocols and sources and that may fall into a permitted traffic class. In most cases, its rate limiting capability can effectively block most of the packets belonging to such attacks. However, because it is not always possible to precisely define good and bad traffic, there are circumstances in which some legitimate packets may be discarded as well.

Port-filtering and Queue-thresholding provide further protection, overcoming some of the limitations present with software-based CoPP. Unlike CoPP, Port-filtering provides a mechanism to dynamically block or permit packets from connections dynamically negotiated, which helps prevent attacks that attempt to spoof legitimate established connections. Routers use protocols such as FTP that dynamically open and close ports. In an ideal scenario, traffic destined to these ports should be allowed while the connections remain open, and should be blocked when the connections are ended. Another limitation of software-based CoPP is that a single class can match traffic from various protocols, and nothing prevents the entire class to be overwhelmed by a flood of traffic matching a single protocol. Queue-thresholding limits the number of packets per protocol so no single misbehaving protocol can jam an entire traffic class.

Control Plane Protection Deployment Recommendations

By protecting the RP, Control Plane Protection helps ensure router and ultimately network stability during an attack. For this reason, Control Plane Protection should be deployed on all routers as a key protection mechanism.

Because Control Plane Protection filters traffic, it is critical to gain an adequate level of understanding about the legitimate traffic destined to the RP before deployment. Control Plane Protection policies built without proper understanding of the protocols, devices or required traffic rates involved may block critical traffic. This has the potential of creating a denial of service (DoS) condition. Determining the exact traffic profile needed to build the Control Plane Protection policies might be difficult in some networks. For this reason, this document describes a conservative methodology for deploying Control Plane Protection using iterative ACL configurations to help identify and to incrementally filter traffic.

Design Considerations and Limitations

The following are some design considerations and limitations to keep in mind when deploying Control Plane Protection:

- Existing (aggregate) control plane policing policies are not affected when the Control Plane Protection functionality is enabled.
- Control Plane Protection does not currently provide support for distributed or hardware switching platforms.

- Control Plane Protection is restricted to IPv4 input path only.
- The current release of Control Plane Protection does not support direct access control list (ACL) configuration in the control-plane subinterfaces, but rather can be configured using Modular QoS CLI (MQC) policies.
- Control Plane Protection depends on Cisco Express Forwarding (CEF) for IP packet redirection. Disabling CEF globally removes all active protect and policing policies configured on the control plane subinterfaces. Aggregate control plane interface policies will continue to function as normal.
- Port-filter and Queue-thresholding are features applicable only to the control plane host subinterface and support only TCP/UDP-based protocols.
- The control plane host subinterface supports only TCP/UDP-based host traffic. All IP packets entering the control plane with IP options or TTLS less than or equal to 1 are not classified any further and are redirected to the CEF-exception subinterface.
- Some Cisco IOS TCP/UDP-based services, when configured, may not be auto-detected by the port-filter. That is, they do not get listed under the show control plane host open ports output and they are not classified as an open port. This type of port must be manually added to the active port-filter class-map to be unblocked.
- There are no restrictions on existing aggregate control-plane policing policies. New control plane policing policies that are configured on host subinterface do not process ARP traffic because ARP traffic is processed at the CEF-exception and aggregate interfaces.

Platform and Software Availability

Control Plane Protection has been introduced in 12.4(4)T for platforms that support software-based CoPP. This feature is not currently supported on distributed or hardware-switching platforms.

Dependencies and Prerequisites

The following are the dependencies and prerequisites required for implementing CoPP:

- Enabling Cisco Express Forwarding (CEF)
- Understanding the principles of Control-plane Policing and how to classify control-plane traffic
- Understanding the concepts and general configuration procedure (class map and policy map) for applying QoS policies on a router

Control Plane Protection Sample Configuration

Defining Control Plane Protection policies requires the use of the MQC CLI following the same procedures explained for Control Plane Policing. For examples illustrating the use of MQC CLI to configure traffic classes and policies, see the Control Plane Policing section of this document.

Assuming that a control plane protection has been configured previously using MQC CLI, the following example shows how the policy is applied to the control plane host subinterface:

```
Router(config)# control-plane host
Router(config-cp)# service-policy input copp-policy
```

The following example shows how to configure a port-filter policy to drop all traffic destined to closed or "nonlistened" TCP/UDP ports:

```
Router(config)# class-map type port-filter pf-class
Router(config-cmap)# match closed-ports
Router(config-cmap)# exit
Router(config)# policy-map type port-filter pf-policy
Router(config-pmap)# class pf-class
Router(config-pmap-c)# drop
Router(config-pmap-c)# end
Router#
```

The following example shows how to configure a queue-threshold policy to set the queue limit for SNMP protocol traffic to 50, telnet traffic to 50, and all other protocols to 150:

```
Router(config)# class-map type queue-threshold qt-snmp-class
Router(config-cmap)# match protocol snmp
Router(config-cmap)# class-map type queue-threshold qt-telnet-class
Router(config-cmap)# match protocol telnet
Router(config-cmap)# class-map type queue-threshold qt-other-class
Router(config-cmap)# match host-protocols
Router(config-cmap)# exit
Router(config)# policy-map type queue-threshold qt-policy
Router(config-pmap)# class qt-snmp-class
Router(config-pmap-c)# queue-limit 50
Router(config-pmap-c)# class qt-telnet-class
Router(config-pmap-c)# queue-limit 50
Router(config-pmap-c)# class qt-other-class
Router(config-pmap-c)# queue-limit 150
Router(config-pmap-c)# end
Router#
```

Useful Debugs and Show Commands

The **show policy-map control-plane** command is a valuable tool for verifying Control Plane Protection configurations and viewing statistics for control-plane service policies. 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 is a sample output from the **show policy-map control-plane** command:

```
Router# show policy-map control-plane transit

 control-plane Transit

  Service-policy input: copp-transit-policy

    Class-map: copp-transit-class (match-all)
      8 packets, 592 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      police:
          rate 2000 pps, burst 488 packets
        conformed 8 packets; actions:
          transmit
        exceeded 0 packets; actions:
          drop
        conformed 0 pps, exceed 0 pps

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Management and Telemetry

The CBQoSMIB, the primary accounting mechanism for MQC-based policies, provides SNMP MIB support for monitoring and managing Control Plane Protection. This functionality is available from Cisco IOS Software Releases 12.2(5)S and 12.3(7)T.

For additional information about CBQoSMIB, see the following website:

ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CLASS-BASED-QOS-MIB-CAPABILITY.my

For additional information about the CBQoSMIB configuration and statistics, see the following website:

ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CLASS-BASED-QOS-MIB.my

Related Tools

Control Plane Protection is a feature that extends Control Plane Policing (CoPP) in platforms that do not support the hardware-based or distributed version of this feature. Although there are no direct dependencies between this two features, when deployed together they build a dual layer of protection that effectively shields the control plane from direct attacks and distributed DoS attacks. In addition, Control Plane Protection Port-filter and Queue-thresholding provide advanced protection by overcoming some of the limitations present with software-based CoPP.

For more information about Control Plane Policing, see Control Plane Policing, page 21.

The Control Plane Protection feature is bundled with a logging feature called Control Plane Logging. Control Plane Logging enables the logging of packets dropped or permitted by Control Plane Protection and associated features. Logging can be enabled for all or some packets that are processed by the control plane, or for specific Control Plane Protection features such as Control Plane Policing, Port-filtering, and Queue-thresholding.

For additional information about the Control Plane Logging, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124newft/124t/124t6/ht_cpl.htm

More Information

For more information about Control Plane Protection, see the following website:

http://www.cisco.com/en/US/partner/products/ps6441/products_feature_guide09186a0080556710.html

Integrated Deployment Guidelines

The tools and techniques described in this document are very valuable for protecting Cisco IOS software-based platforms from direct attacks as well as the negative effects of accidental misconfiguration. Even though most of the tools here described work independently, they are also complementary and provide even greater value when deployed in an integrated way. This section describes the interrelations between these tools and provides general guidelines for deploying these tools as an integrated solution, rather than as isolated elements. This section includes the following topics:

- Basic Tools and Techniques for Infrastructure Protection
- Infrastructure Protection Access Control Lists
- rACLs and CoPP

- CoPP and Control Plane Protection

Basic Tools and Techniques for Infrastructure Protection

The "Basic Tools and Techniques for Infrastructure Protection" section on page 3 describes a collection of features and techniques that together compose an essential toolkit for securing Cisco IOS software-based devices. The deployment of these tools on routers and switches is recommended for most environments.

This toolkit includes features that help prevent indiscriminate consumption of the limited resources on a device. In addition, Appendices A and B provide techniques for disabling unnecessary services, which will save device resources or help control access to the device. All these are essential security services that together provide a security baseline.

Enforcing proper device access is always recommended. The following are best practices that should be followed whenever it is possible:

- Adequate password management,
- Controlling console and interactive access
- Deploying banners
- Enforcing role-based access

These practices are described in Appendix A.

Likewise, tuning input hold queues, implementing ICMP unreachable overload protection, and configuring the Scheduler provide basic mechanisms for protecting the device itself from indiscriminate resource consumption.

Disabling unnecessary services is another general best practice. Routers and switches often run a collection of services by default, which could potentially be used maliciously. Disabling all unnecessary global and interface services greatly reduces the risk of security incidents. In cases where a service is still needed, it should be selectively deployed. The service may not be needed globally, but only on specific interfaces.

Directed broadcasts are a good example of a service that, as a general best practice, should never be globally enabled, but that under some particular circumstances might still be needed. For example, messaging middleware, such as TIBCO Info Mediator, may rely on directed broadcasts for system communication.

However, directed broadcast should not be enabled indiscriminately on all interfaces because this clearly increases the likelihood of security incidents, such as SMURF attacks. On the contrary, directed broadcasts should only be enabled on those interfaces that have systems communicating with TIBCO Info Mediator or other essential services requiring directed broadcasts. In addition, and to properly secure this type of environment, ACLs need to be deployed on all interfaces to make sure only expected sources send directed broadcasts to the specific interfaces enabled to accept them.

Infrastructure Protection Access Control Lists

Infrastructure Protection ACLs are filters designed to protect the infrastructure from direct attacks. These filters restrict external access to the infrastructure address space, allowing only authorized devices to communicate with infrastructure elements like routers and switches, while allowing transit traffic to flow freely. iACLs represent a first line of protection against external threats, so they should be deployed at network ingress points. More precisely, iACLs should be implemented on the ingress interfaces of the network devices that provide the first line of access to the network.

Overall, there are no dependencies between iACLs and other filtering techniques described in this document. The only special consideration is that any additional filtering deployed on the ingress devices such as CoPP or rACLs should be aligned with the filters enforced by iACLs. As described in the "Infrastructure Protection Access Control Lists" section on page 5, Module 2 of an iACL contains the explicit **permit** statements that allow external access to the infrastructure address space from authorized sources and for the expected protocols. The iACLs rules in Module 2 that allow traffic to the control plane of the ingress device should have corresponding entries in the CoPP or rACL configuration.

This is illustrated in the following example, where Module 2 of the iACL has specific permit rules allowing a peering router (169.223.253.2) to maintain a BGP session with an ingress router (169.223.253.1):

```
….
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Module 2 of iACL:  Explicit Permit
!--- Permit external BGP.
access-list 110 permit tcp host 169.223.253.2 host 169.223.253.1 eq bgp
access-list 110 permit tcp host 169.223.253.2 eq bgp host 169.223.253.1
!
…
```

This iACL is applied on the external facing interfaces of this ingress router.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- iACL applied to an external facing interface
interface FastEthernet0/0
 ip access-group 110 in
```

CoPP should be configured to allow this control plane traffic. To that end, this BGP traffic is associated with the BGP traffic class, which in this example is allowed without any rate limits.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Fragment of basic ACLs for CoPP classification
!
! In this network, the peering BGP session is defined as part
! of the BGP class
ip access-list extended coppacl-bgp
 remark BGP traffic class
! allow BGP from a known peer to this router's BGP TCP port
permit tcp host 169.223.253.2 host 169.223.253.1 eq bgp
! allow BGP from a peer's BGP port to this router
permit tcp host 169169.223.253.2 eq bgp host 169.223.253.1
….
! Policy configuration for the BGP class
class-map match-all coppclass-bgp
  match access-group name coppacl-bgp
!
….
! This is the actual policy
! Critical traffic will not be limited in any way therefore no operation needs to be
specified in this class.
policy-map copp-policy
  class coppclass-bgp
…
! Applies the defined CoPP policy to the control plane
Router(config)# control-plane
Router(config-cp)# service-policy input copp-policy
```

rACLs and CoPP

rACLs and CoPP are both features that help protect the RP from unnecessary traffic destined to the control plane and that could potentially affect system performance. For that reason, it is always a good practice to deploy CoPP, or rACLs when CoPP is not available, on all Cisco IOS software-based routers and switches. That helps maintain network stability even during network attacks.

Overall, CoPP is preferable to rACLs because it is more effective and allows for a more flexible policy. As discussed in the "Control Plane Policing" section on page 21, rACLs can only block or permit traffic, while CoPP can implement flexible rate limiting policies that better fit real-world scenarios. Distributed CoPP provides even greater flexibility in policy definition. Distributed CoPP should be deployed whenever it is available and whenever distinctive policies are needed at the line-card level.

In terms of configuring rACLs and CoPP, when either of these features are deployed in conjunction with iACLs, the rules defined through CoPP or rACLs should be aligned with the filters enforced by the iACL. This situation only occurs on the ingress routers and switches, because those are the only places where iACLs should be deployed.

If rACLs have already been deployed, it is recommended that you migrate them to CoPP whenever this feature becomes available. This section describes the recommended migration methodology and provides an example for migrating rACLs to CoPP.

CoPP Migration Methodology

This section describes a conservative approach for the migration of rACLs policies to CoPP. This approach starts with an initial configuration that emulates the actual rACL deployment, and that gradually refines the policies by implementing rate limiting. To implement this methodology, complete the following steps:

Step 1 Determine the classification scheme for your network.

Identify the known protocols that access the RP and divide them into categories. As described in the "CoPP Deployment Recommendations" section on page 28, you can use various classification schemes, including a combination of relative importance and traffic type (for example, BGP, IGP, Management, Reporting, Monitoring, Critical Applications, Layer 2 Protocols, Undesirable, and Default).

Select a scheme suited to the specific network environment, which may require a larger or smaller number of classes.

Step 2 Match rACLs entries to the CoPP traffic classes.

a. Create an ACL for each CoPP traffic class.

b. Map each **permit** entry on the rACL to the ACL that corresponds to their class.

To do this, take each **permit** entry on the rACL and add a corresponding **permit** entry on the ACL representing the traffic class.

c. Map any **deny** entry that occurs in the rACL to the Undesirable class or equivalent.

To do this, take each **deny** entry on the rACL and add a **permit** entry with the same parameters.

Note Because you are defining classification ACLs, even an ACL that represents an Undesirable traffic class should be constructed using **permit** statements.

Step 3 Emulate rACL policies with CoPP configuration.

a. Configure a CoPP policy for each traffic class to either transmit or drop traffic. A policy configuration using only **transmit** and **drop** actions is analogous to an rACL. As a reference, all traffic classes built from **permit** entries on the original rACL should be configured with a **transmit** policy. Likewise, all traffic classes built from **deny** entries on the rACL should be set to **drop**.

b. (Optional) Define a rate limit policy for traffic falling into the Default class. Give this traffic a very low limit.

c. Disable the rACL.

d. Apply the newly defined CoPP policies to the control plane.

Step 4 Refine CoPP 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.

CoPP Migration Example

Step 1 Determine the classification scheme for your network.

In this example, the control plane traffic is classified into seven classes, each of which is associated with a separate extended ACL:

- BGP (coppacl-bgp)—BGP traffic
- IGP (coppacl-igp)—OSPF traffic
- Interactive Management (coppacl-interactivemanagement)—Remote access and management traffic such as TACACS, SSH, SNMP, and NTP.
- File Management (coppacl-filemanagement)—Remote file transfer traffic such as TFTP and FTP.
- Monitoring (coppacl-monitoring)—ICMP and traceroute traffic
- Undesirable Traffic (coppacl-undesirable)—Explicitly denies unwanted traffic (for example, Slammer Worm packets)
- Default (no ACL needed)—All traffic received by the control plane that has not been otherwise identified

Step 2 Match rACLs entries to the CoPP traffic classes.

One ACL is created per class. For each entry on the rACL, a corresponding line is added in the ACLs corresponding to each class.

```
! In this example, BGP must be defined in a separate class
ip access-list extended coppacl-bgp
 remark BGP traffic class
 permit tcp 169.223.252.0 0.0.3.255 host 169.223.253.1 eq bgp
 permit tcp 169.223.252.0 0.0.3.255 eq bgp host 169.223.253.1
!
! OSPF should be defined as part of the IGP class
ip access-list extended coppacl-igp
 remark IGP traffic class
! Permit OSPF
 permit ospf 169.223.252.0 0.0.3.255 host 224.0.0.5
 permit ospf 169.223.252.0 0.0.3.255 host 224.0.0.6
 permit ospf 169.223.252.0 0.0.3.255 host 169.223.253.1
!
```

```
! Telnet, ssh, SNMP, NTP and Tacacs+ traffic should be
! assigned to the Interactive Management class
ip access-list extended coppacl-interactivemanagement
 remark CoPP interactive management traffic class
! Remote access: telnet and ssh
 permit tcp 172.16.1.0 0.0.0.255 host 169.223.253.1 eq 22
 permit tcp 172.16.1.0 0.0.0.255 host 169.223.253.1 eq telnet
! SNMP
 permit udp 172.16.1.0 0.0.0.255 host 169.223.253.1 eq snmp
! NTP
 permit udp 172.16.1.0 0.0.0.255 host 169.223.253.1 eq ntp
! TACACS for router authentication
 permit tcp 172.16.1.0 0.0.0.255 host 169.223.253.1 established
!
! TFTP and FTP traffic is classified in the File Management class
ip access-list extended coppacl-filemanagement
 remark CoPP file management traffic class
! Allow router initiated FTP (active and passive)
 permit tcp 172.16.1.0 0.0.0.255 eq 21 host 169.223.253.1 gt 1023 established
 permit tcp 172.16.1.0 0.0.0.255 eq 20 host 169.223.253.1 gt 1023
 permit tcp 172.16.1.0 0.0.0.255 gt 1023 host 169.223.253.1 gt 1023 established
! Allow router initiated TFTP
 permit udp 172.16.1.0 0.0.0.255 gt 1023 host 169.223.253.1 gt 1023

! In this classification scheme, 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
! Router originated traceroute
! Each hop returns a ttl exceeded (type 11, code 3) message
! and the final destination returns an ICMP port unreachable
! (type 3, code 0)
 permit icmp any 169.223.253.1 ttl-exceeded
 permit icmp any 169.223.253.1 port-unreachable
!
! This ACL identifies traffic that should always be blocked from accessing the Route
Processor.
ip access-list extended coppacl-undesirable
 remark explicitly defined "undesirable" traffic
! Denies all traffic destined to UDP 1434 (Slammer), optionally count denied packets for
tracking
 deny udp any any eq 1434
!
```

Step 3 Emulate rACL policies with CoPP configuration.

Each class is configured with a **transmit** or **drop** action. Classes BGP, IGP, interactive management, file management, and monitoring are permitted without limits. In addition, a Default class is defined with a low rate limit.

```
! Defines a class for each "type" of traffic and associates it with an 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-interactivemanagement
  match access-group name coppacl-interactivemanagement
class-map match-all coppclass-filemanagement
  match access-group name coppacl-filemanagement
class-map match-all coppclass-monitoring
  match access-group name coppacl-monitoring
class-map match-all coppclass-undesirable
  match access-group name coppacl-undesirable
!
```

```
! This is the actual policy. In this base case, all actions are transmit actions except
for
! traffic defined as undesirable which is unconditionally dropped regardless of rate and
the
! default class that is rate limited above the specified bps rate.
policy-map copp-policy
  class coppclass-bgp
     police 50000 1500 1500 conform-action transmit  exceed-action transmit
  class coppclass-igp
   !<no operation specified since this class has unrestricted access to Route Processor>
  class coppclass-interactivemanagement
     police 50000 1500 1500 conform-action transmit  exceed-action transmit
  class coppclass-filemanagement
   !<no operation specified since this class has unrestricted access to Route Processor>
  class coppclass-monitoring
     police 50000 1500 1500 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 50000 1500 1500 conform-action drop  exceed-action 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 10,000,000
bps and
   ! violations of that limit are dropped.
   class class-default
     police 500000 5000 5000 conform-action transmit  exceed-action drop

The rACL is removed and the newly CoPP is applied to the control plane

Router# configure terminal
Router(config)# no ip receive access-list 110
Router(config)# control-plane
Router(config-cp)# service-policy input copp-policy
```

Step 4 Refine CoPP policies by implementing rate limiting.

After periodically analyzing the packet count and rate information provided with the **show policy-map control-plane** command, a more redefined set of policies is enforced.

```
! IGP and File Management traffic will not be limited in any way therefore no operation
needs
! to be specified in this class.
policy-map cpp
class coppclass-igp
class coppclass-filemanagement
! BGP traffic is limited to a rate of 80,000 bps, if traffic exceeds
! that rate it is dropped.
  class coppclass-bgp
     police 80000 8000 8000 conform-action transmit exceed-action drop
! Interactive Management traffic is limited to a rate of 10,000,000 bps, if traffic
! exceeds that rate it is dropped
  class coppclass-interactivemanagement
    police 10000000 100000 100000 conform-action transmit  exceed-action drop
 ! Monitoring traffic is limited to a rate of 500,000 bps, if traffic exceeds
 ! that rate it is dropped
  class coppclass-monitoring
    police 500000 5000 5000 conform-action transmit  exceed-action drop
  ! This policy drops all traffic categorized as undesirable, regardless of rate.
  class coppclass-undesirable
     police 50000 1500 1500 conform-action drop exceed-action 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 10,000,000
bps and
  ! violations of that limit are dropped.
  class class-default
     police 10000000 100000 100000 conform-action transmit exceed-action drop
```

CoPP and Control Plane Protection

Control Plane Protection and Control Plane Policing (CoPP) are two independent security features; in fact, when deployed together, existing CoPP policies are not affected by Control Plane Protection. However, even though there are no direct dependencies between these two features, better results are obtained when used together. As described in Control Plane Protection, page 42, Control Plane Protection extends the policing functionality of CoPP by providing an additional layer of protection to the control plane. In addition, Port-filtering and Queue-thresholding provide enhanced protection by overcoming some of the limitations present in software-based CoPP. For these reasons, deploying CoPP in conjunction with Control Plane Protection maximizes the protection of the control plane.

Deploying CoPP with Control Plane Protection creates a dual layer protection hierarchy. It is a good practice to define traffic policies that take advantage of this hierarchy. For example, a single high level policy can be defined at the aggregate CoPP level, while more granular policies can be set at each control plane protection subinterface. In this way, the aggregate CoPP policy establishes the rate limits for all traffic destined to the control plane, while the individual policies at each control plane subinterface enforce finer limits per traffic type.

Appendix A—Turning Off Unnecessary Services

To facilitate deployment, Cisco routers and switches come out of the box with a list of services turned on that are appropriate for many network environments. However, the infrastructure networks described in this document have specialized requirements in which many of these services are not needed and can be disabled. Disabling these unnecessary services has two benefits. It helps preserve system resources, and it eliminates the potential of security exploits using the disabled services.

This appendix describes how to disable some services that may not be needed. As an alternative, Cisco IOS software provides the **AutoSecure** CLI command that helps disable these unnecessary services, while enabling other security services.

Note Before disabling a service make sure the service is not needed.

This appendix describes the procedure for disabling the following services, which are typically not needed in infrastructure networks:

- Cisco Discovery Protocol (CDP)
- Directed Broadcast
- Finger
- Maintenance Operations Protocol (MOP)
- HTTP Server
- IP BOOTP Server

- IP Redirects
- IP Source Routing
- PAD
- Proxy ARP
- Ident
- TCP and UDP Small Servers

Cisco Discovery Protocol (CDP)

Cisco Discovery Protocol is a Cisco proprietary Layer 2 protocol designed to facilitate the administration and troubleshooting of network devices by providing information on neighboring equipment. With CDP enabled, network administrators can execute CDP commands that provide them with the platform, model, software version, and even the IP addresses of adjacent equipment.

CDP is a useful protocol, but could clearly reveal important information to an attacker. CDP is enabled by default, and can be disabled globally or for each interface. The best practice is to disable CDP globally when the service is not used, or on each interface when CDP is still required. In cases where CDP is used for troubleshooting, CDP should be left enabled globally, and should be disabled only on those interfaces on which the service may represent a risk, for example, interfaces connecting to the Internet. As a general practice, CDP should not be enabled on interfaces that connect to external networks, such as the Internet.

To disable CDP globally use the **no cdp run** command from global configuration mode, as in the following example:

```
router(config)# no cdp run
```

To disable CDP on one or more interfaces, use the **no cdp enable** command from interface configuration mode, as in the following example:

```
router(config-if)# no cdp enable
```

Note Features such as ODR (on demand routing) depend on CDP, so check for dependencies prior to disabling CDP.

For more information about CDP, see the following website:

http://www.cisco.com/en/US/partner/tech/tk962/technologies_tech_note09186a00801aa000.shtml

Directed Broadcast

An IP directed broadcast packet is an IP packet whose destination address is a valid broadcast address for an IP subnet. When a directed broadcast packet reaches a router that is directly connected to its destination subnet, and if the router is configured to do so, that packet is “exploded” as a broadcast on the destination subnet. By default, earlier releases of Cisco IOS software handle directed broadcasts this way. However, because directed broadcasts have been used for attacks, such as the SMURF attack, the default behavior has been changed to drop directed broadcasts since Cisco IOS software Release 11.2.

In the case the forwarding of directed broadcast has been enabled, or in the case of Cisco IOS software releases prior to Cisco IOS software Release 11.2, it is s recommended that you disable this feature on all interfaces using the **no ip directed-broadcast**. interface configuration command, as in the following example:

```
router(config-if)# no ip directed-broadcast
```

For more information about the **ip directed-broadcast** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/ipras_r/ip1_i1g.htm#wp1081245

Finger

Finger, as defined in RFC 742, is a protocol that can be used to obtain information about users logged into a remote host or network device. Cisco IOS software incorporates a finger service, which in Cisco IOS software releases prior to 12.1(5) and 12.1(5)T was turned on by default. Although the finger service does not reveal any extremely sensitive information, it can be used by a potential attacker to gather information Therefore it is recommended that you disable this service.

In older releases of Cisco IOS software where the finger service was enabled by default, it can be disabled with the **no service finger** global configuration command, as in the following example:

```
router(config)# no service finger
```

Starting in Cisco IOS software 12.1(5) and 12.1(5)T, the finger service is disabled by default. If finger has been turned on and the service is not needed, it can be disabled with the **no ip finger** global configuration command, as in the following example:

```
router(config)# no ip finger
```

For more information on the finger service, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g03.htm#wp1033299

Maintenance Operations Protocol (MOP)

The Maintenance Operations Protocol (MOP) was developed by Digital Equipment Corporation to be used for remote communications between hosts and servers. Cisco IOS software routers implement MOP to gather configuration information when communicating with DECNet networks. By default, MOP is enabled on all Ethernet interfaces, and disabled on all other type of interfaces. The MOP service can be disabled per interface by using the **no mop enabled** interface configuration command, as in the following example:

```
router(config-if)# no mop enabled
```

MOP has been proven vulnerable to various attacks; therefore it should be disabled on all interfaces unless they provide connectivity to DECNet networks.

For more information about the **mop enabled** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/inter_r/int_m1g.htm#wp1139514

HTTP Server

Introduced in Cisco IOS software Release 11.2, the Cisco IOS software Web browser user interface allows the configuration of Cisco IOS software-based devices by using a web browser such as Internet Explorer or Netscape. This user interface relies on a built-in HTTP server service that runs on Cisco IOS software and is turned off by default. Because of the nature of HTTP this service does not provide encryption for client connections, which leaves communication between clients and servers vulnerable to interception and attack. In case the HTTP server has been enabled and is not needed, it is recommended to turn it off using the **no ip http server** global configuration command, as in the following example:

```
router(config)# no ip http server
```

If a web interface is required, you should consider the following options, described in this section, to improve security:

- Deploy HTTPS, a secure version of HTTP
- Implement access controls on the HTTP server
- Change the default user name and password for the Security Device Manager (SDM) web interface

HTTPS

The Secure HTTP (HTTPS) feature provides the capability to connect to the Cisco IOS software web server securely. It uses Secure Sockets Layer (SSL) and Transport Layer Security (TLS) to provide device authentication and data encryption.

This feature was introduced on 7100 series routers and 7200 series routers in Cisco IOS software Release 12.1(11b)E, and integrated into 12.2(14)S. HTTPS with SSL version 3.0 was integrated into 12.2(15)T for all platforms supporting Cisco IOS software software images with SSL support (IPSec 56 and IPSec 3DES images).

The HTTPS server can be enabled with the **ip http secure-server** global configuration command, as in the following example:

```
router(config)# ip http secure-server
```

For more information, see the following website:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps1833/products_feature_guide09186a00800d9eee.html#wp1021949

For more information about the HTTP server and the HTTP client with SSL 3.0, see the following website:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps1839/products_feature_guide09186a008015a4c6.html

HTTP Server Access Controls

Access to the HTTP Server can be controlled by enabling AAA user authentication with the **ip http authentication** global command, as in the following example:

```
router(config)# ip http authentication
```

User authentication can be implemented using a local user list, or using protocols like TACACS+ or RADIUS. The following example shows a configuration listing for HTTP authentication using TACACS+.

```
aaa new-model
aaa authentication login default group tacacs+
aaa authorization exec default group tacacs+
ip http server
ip http authentication aaa
tacacs-server host 171.68.18.10
tacacs-server key cisco
```

For more information about HTTP authentication see the following website:

http://www.cisco.com/en/US/partner/tech/tk59/technologies_configuration_example09186a0080178a51.shtml

For information about using a non-standard port for HTTP (other than 80) with the **ip http port** global command, see the following website

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g04.htm#wp1028992

You can also allow only trusted hosts or networks to access the HTTP server by using the **ip http access-class** global configuration command. The following example shows how access to the HTTP server is configured to only be allowed from a single host (10.0.0.1).

```
router(config)# ip http access-class 10
router(config)# access-list 10 permit host 10.0.0.1
```

For more details about the **http access-class** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g04.htm#wp1028455

You can limit the maximum number of connections to the HTTP server using the **ip http max-connections** global configuration command, as in the following example

```
router(config)# ip http max-connections 3
```

This limits the maximum number of concurrent connections to three.

For more information, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g04.htm#wp1028838

Security Device Manager (SDM) – Special Considerations

Security Device Manager (SDM) is an embedded Web-based management tool that is pre-installed on some Cisco IOS software-based platforms and that comes with a factory default login configuration. The factory configuration enables the HTTPS server service, and sets a default user name and password. For obvious security reasons, it is highly recommended that you change this default login configuration prior to deploying the device, by either using the CLI or the SDM configuration wizard.

For more information about SDM, see the following website:

http://www.cisco.com/en/US/partner/products/sw/secursw/ps5318/index.html

IP BOOTP Server

As defined by RFC 951, the Bootstrap protocol allows a diskless workstation to configure itself at boot time by dynamically obtaining an IP address, the IP address of the BOOTP server, and a configuration file. Cisco IOS software implements a bootstrap service that allows a router to act as a BOOTP server

providing dynamic configuration services to other Cisco IOS software routers. This service is turned on by default and it is used by features like AutoInstall, which simplifies or automates the configuration of Cisco devices. If not needed, this service should be disabled with the **no ip bootp server** global configuration command, as in the following example;

```
router(config)# no ip bootp server
```

For more information about the BOOTP server service, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g03.htm#wp1031545

For more information about AutoInstall, see the following website:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800ca735.html

IP Redirects

By default, Cisco IOS software sends ICMP redirect messages when it is forced to resend a packet through the same interface on which it was received. By sending these redirect messages the router instructs the host the specific router to use to reach a particular destination. The ICMP redirect messages can also reveal information that can potentially be used by an attacker for discovering the network topology. Therefore, it is highly recommend that you disable this service on all interfaces. IP redirects can be disabled on each interface using the **no ip redirects** interface configuration command, as in the following example:

```
router(config-if)# no ip redirects
```

For more information about the **ip redirect** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/ipras_r/ip1_i2g.htm#wp1081518

Note Previously, if the Hot Standby Router Protocol (HSRP) was configured on an interface, ICMP redirect messages were disabled by default for the interface. With Cisco IOS software Release 12.1(3)T, ICMP redirect messages are enabled by default even if HSRP is configured.

IP Source Routing

The IP protocol supports source routing options that allow the sender of an IP packet to control the route that the datagram will take toward its ultimate destination, and generally the route that any reply will take. These options are rarely used for legitimate purposes in real networks. Some older IP implementations do not process source-routed packets properly, and it may be possible to crash machines running these implementations by sending them datagrams with source routing options. By default Cisco IOS software forwards IP packets with source routing header options. As a general best practice, IP source routing should be disabled unless strictly necessary. To have the software discard any IP packet containing a source-route option, use the **no ip source-route** global configuration command as in the following example:

```
router(config)# no ip source-route
```

For more information about the **ip source-route** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/ipras_r/ip1_i2g.htm#wp1081830

PAD

Cisco IOS software provides a PAD (packet assembler/disassembler) service that allows simple devices such as character-mode terminals to connect to legacy X.25 networks. With this service Cisco IOS software devices and other X.25 network equipment can establish PAD sessions. By default, the PAD service is enabled on Cisco IOS software, but it could be used to gain unauthorized or inappropriate access. Therefore, unless needed, this service should be disabled with the **no service pad** global configuration command, as in the following example:

```
router(config)# no service pad
```

For more information about the PAD service see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/wan_r/wan_s1g.htm#wp1032441

Proxy ARP

Proxy Address Resolution Protocol (ARP), as defined in RFC 1027, is a technique that helps machines on a subnet reach remote subnets without configuring routing or a default gateway. Proxy ARP is typically implemented on routers, and when configured, the router answers all ARP requests on the local subnet on behalf of systems some hops away.

In this model, local hosts send ARP requests for each destination for which they do not have any routing information, and the router replies with its own MAC address as the next hop. Cisco IOS software by default implements proxy ARP on all interfaces. However, unless strictly needed it should be disabled with the **no ip proxy-arp** interface configuration command, as in the following example:

```
router(config-if)# no ip proxy-arp
```

For more information about the **ip proxy-arp** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/ipras_r/ip1_i2g.htm#wp1081466

Ident

As defined by RFC 1413, the TCP Client Identity Protocol (Ident) is a protocol that allows a system to query the identity of a user initiating a TCP connection or a host responding to a TCP connection. When implemented, the Ident service allows a user to obtain identity information by simple connecting to a TCP port on a system, and issuing a simple text string requesting information. This clearly can yield information that could be used to attack the system. Cisco IOS software routers implement an Ident service, which is disabled by default. It is highly recommended that you do not enable this service. If the Ident service has been enabled, it can be disabled by using the **no ip identd** global configuration command, as in the following example:

```
router(config)# no ip identd
```

For more information about the **ip identd** command, see the following website:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps1828/products_command_reference_chapter09186a00800ca5e6.html#wp1017578

TCP and UDP Small Servers

TCP and UDP small servers are daemons that typically run on Unix systems and that were designed for diagnostic purposes. Cisco IOS software also provides an implementation of UDP and TCP small servers that enables echo, chargen, daytime and discard services. Unless strictly necessary, these services should be disabled because they can be used by a potential attacker to gather information, or to directly attack the Cisco IOS software device.

TCP and UDP small services are enabled by default on Cisco IOS software Release 11.2 and earlier. These commands are disabled by default on Cisco IOS software Software Versions 11.3 and later.

These commands may be disabled using the **no service tcp-small-servers** and **no service udp-small-servers** global configuration commands, as shown in the following example:

```
router(config)# no service tcp-small-servers
router(config)# no service udp-small-servers
```

For more information about TCP and UDP small servers, see the following website:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps1818/products_tech_note09186a008019d97a.shtml#tcp_udp_servers

Appendix B—Controlling Device Access

There are more access mechanisms to an Cisco IOS software device than many administrators realize, from console and asynchronous connections, to remote sessions based on protocols like Telnet, rlogin and SSH. Most of these mechanisms are not enabled by default, but others like console and access from integrated modem lines are. In every cases it is critical to control who accesses the device. Anyone who gains access to a router or switch may obtain critical information about the network, reconfigure the device, and even take the device out of service. For this reason, the device should be carefully configured to prevent any unauthorized access. This section describes the following best practices, which help control access to Cisco IOS software-based equipment:

- Password management
- Interactive access control
- Banners
- Role-based CLI access

Password Management

Passwords (and similar secrets, such as SNMP community strings) are the primary defense against unauthorized access to your router. The best way to handle most passwords is to maintain them on a TACACS+ or RADIUS authentication server. However, almost every router and switch will still have a locally configured password for privileged access, and may also have other password information in its configuration file. The following paragraphs describe some of the CLI commands available on the device to help prevent unauthorized access.

enable secret

The **enable secret** global command is used to set the password that grants privileged administrative access to the Cisco IOS software system. By default, no enable secret password is enabled, and as a general best practice, a password that will resist a simple dictionary attack should always be set .

To set an enable secret password, use the **enable secret** global configuration command, as in the following example:

```
router(config)# enable secret Hard2Guess
```

Passwords should be at least eight characters in length, with a combination of number, and upper-case letters.

Cisco IOS software also offers the older **enable password** command, but it is not recommended because it uses a weak encryption algorithm. The **enable secret** command provides stronger encryption based on MD5 hashing.

In addition, if no enable secret is set, and a password is configured for the console TTY line, the console password may be used to get privileged access, even from a remote VTY session. This is not a recommended practice, and makes for another good reason to configure an enable secret.

For more information about the **enable secret** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/secur_r/sec_d1g.htm#wp1070932

service password-encryption

By default, some passwords and secrets are shown in clear text in a Cisco IOS software configuration file or listing. The **service password-encryption** global configuration command instructs Cisco IOS software to encrypt the passwords, CHAP secrets, and similar data that are saved in the configuration file. This is shown in the following example:

```
router(config)# service password-encryption
```

This command is primarily useful for keeping unauthorized individuals from viewing passwords in the configuration file. However, it is important to note that the algorithm used by service password-encryption is a simple Vigenere cipher that can be easily reversed, and for that reason this command should not be used with the intention to protect configuration files against serious attacks.

For more information about the **service password-encryption** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/secur_r/sec_r1g.htm#wp1070450

security password min-length

Introduced in Cisco IOS software Release 12.3(1), the **security password min-length** global configuration command provides enhanced security access to the router by allowing the user to specify a minimum password length, eliminating common passwords that are prevalent on most networks, such as “lab” and “cisco”. For example:

```
router(config)# security password min-length
```

This command affects user passwords, enable passwords and secrets, and line passwords. After this command is enabled, any password that is less than the specified length will fail. This command was also integrated into Cisco IOS software Release 12.2(18)T.

For more information about the **security password min-length** command, see the following website:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps5187/products_feature_guide09186a008017d101.html#wp1048684

Interactive Access Control

Cisco IOS software supports management connections using a diverse set of mechanisms, including Telnet, rlogin, SSH, non-IP-based network protocols like LAT, MOP, X.29, and V.120, as well as using local asynchronous connections and modem dial-ins. In addition, interactive Telnet access is available not only on the standard Telnet TCP port (port 23), but on a variety of higher-numbered ports as well.

In Cisco IOS software, all interactive access mechanisms involve sessions or lines. Local asynchronous terminals and dialup modems use standard lines, known as TTYs. Remote network connections, regardless of the protocol, use virtual TTYs (VTYs).

The best way to protect a system is to make certain that appropriate controls are applied on all lines, including both VTY lines and TTY lines. Because it is difficult to be certain that all possible modes of access have been blocked, administrators should make sure that logins on all lines are controlled using some sort of authentication mechanism, even on machines that are supposed to be inaccessible from untrusted networks. This is especially important for VTY lines and for lines connected to modems or other remote access devices. As mentioned previously, the best way to use passwords and authentication is by deploying protocols like TACACS+, RADIUS or Kerberos.

For more information about how to configure these protocols, see the following website:

http://www.cisco.com/en/US/tech/tk59/tech_protocols_list.html

If not required, interactive logins may be completely disabled on any line by configuring the **login** and **no password** commands at the line configuration level. This is the default configuration for VTYs, but not for TTYs.

Controlling TTYs

Prior to Cisco IOS software Release 11.1, all asynchronous ports (TTY lines) were configured with the **transport input all** command by default, allowing all type of connections to the TTY lines. This configuration allowed remote users to establish connections to TTY lines over the network with a reverse Telnet and interact with the terminals or modems connected to the TTY lines.

Sometimes it is necessary to allow users to make this sort of connection. However, this feature may enable an attacker to connect to a local asynchronous terminal port, or even to a dial-in modem port. The attacker may then simulate the router login prompt to steal passwords, or perform other exploits.

On devices running Cisco IOS software releases earlier than Cisco IOS software Release 11.1, it is highly recommended that, unless strictly needed, the reverse Telnet feature be disabled by entering the **transport input none** configuration command to the asynchronous or modem lines.

Starting in Cisco IOS software Release 11.1, **transport input none** is the default configuration for all TTY lines. By default, no network connections are allowed unless an incoming transport protocol or all the protocols (**transport input all**) are specifically enabled.

For more information about the **transport input** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/tersv_r/ter_t1g.htm#wp1083564

Controlling VTYs and Ensuring VTY Availability

Like TTYs, all types of connections to VTYs were allowed by default prior to Cisco IOS software Release 11.1. Starting in Cisco IOS software Release 11.1, no connections are permitted to VTY lines, unless an incoming protocol or all the protocols are specified with the **transport input** command.

In case VTY access is required, a VTY should be configured to accept connections only with the protocols actually needed. This is done with the **transport input** command. For example, a VTY that is expected to receive only Telnet sessions would be configured with **transport input telnet**, while a VTY permitting both Telnet and SSH sessions would have **transport input telnet ssh**. The best practice is to prefer encrypted access protocols, such as SSH, over clear text protocols like Telnet.

Another good practice is the use of the **access-class** command to restrict the IP addresses from which the VTY will accept connections. By default, once an access protocol is enabled for a VTY line any host can initiate a connection using that protocol. The **access-class** command defines a list of hosts or networks from which access will be allowed, which prevents unauthorized access from untrusted sources. This practice also helps mitigate a denial-of-service attack on the VTY lines.

Cisco IOS software devices have only a limited number of VTY lines, usually five. When all of the VTYs are in use, no more remote interactive connections can be established and this creates an opportunity for a denial-of-service attack. If an attacker can open remote sessions to all the VTYs on the system, the legitimate administrator may not be able to log in. The attacker does not have to log in to do this; the sessions can simply be left at the login prompt.

One way to protect against this attack is to configure a restrictive **access-class** configuration on the last VTY in the system. The last VTY, usually VTY 4, can be restricted to accept connections only from a single, specific administrative workstation, whereas the other VTYs might accept connections from any address in a corporate network.

Another useful tactic is to decrease the VTY timeouts using the **exec-timeout** command. This prevents an idle session from consuming a VTY indefinitely. By default, a VTY session has a 10-minute timeout. Although the effectiveness of this technique against deliberate attacks is relatively limited, it also provides some protection against sessions accidentally left idle. Similarly, enabling TCP keepalives on incoming connections (with the **service tcp-keepalives-in** command) can help to guard against malicious attacks and orphan sessions caused by remote system crashes. By default, keepalives are not enabled for incoming connections.

The following configuration illustrates the best practices just described. In this configuration ,access for VTY 4 is restricted to only SSH connections coming from the IP address 10.0.0.1. The line timeout is set to 2 minutes and 30 seconds, and tcp keepalives are enabled.

```
service tcp-keepalives-in
access-list 10 permit host 10.0.0.1
line vty 4
     transport input ssh
     access-class 10 in
     exec-timeout 2 30
```

Limiting Authentication Failure Rate

Introduced in Cisco IOS software Release 12.3(1), the **security authentication failure rate** global configuration command provides protection against dictionary attacks. In a dictionary attack, automated software attempts to log in using every word in a dictionary.

The **security authentication failure rate** command allows the user to define a maximum number of consecutive unsuccessful login attempts, after which, device access is locked for a period of 15 seconds. Additionally, this command can be configured to generate a syslog message every time the number of unsuccessful login attempts exceeds the configured threshold rate.

The best practice is to configure a maximum threshold of 3 consecutive unsuccessful login attempts, and to enable the generation of syslog messages, as shown in the following configuration.

```
router(config)# security authentication failure rate 3 log
```

This configuration causes access to the router to be locked for a period of 15 seconds after three unsuccessful login attempts, disabling the dictionary method of attack. In addition to locking access to the router, this configuration causes a log message to be generated after three unsuccessful login attempts, warning the administrator of the unsuccessful login attempts.

For more information about the **security authentication failure rate** global command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/secur_r/sec_r1g.htm#wp1081495

Warning Banners

In some jurisdictions, civil and/or criminal prosecution of attackers who break into your systems is made much easier if you provide a banner informing unauthorized users that their use is in fact unauthorized. In other jurisdictions, you may be forbidden to monitor the activities of even unauthorized users unless you have taken steps to notify them of your intent to do so. One way of providing this notification is to put it into a banner message configured with the Cisco IOS software **banner login** global command.

Legal notification requirements are complex, and vary in each jurisdiction and situation. Even within jurisdictions, legal opinions vary, and this issue should be discussed with your own legal counsel. In cooperation with counsel, you should consider which of the following information should be put into your banner:

- A notice that the system is to be logged in to or used only by specifically authorized personnel, and perhaps information about who may authorize use.
- A notice that any unauthorized use of the system is unlawful, and may be subject to civil and/or criminal penalties.
- A notice that any use of the system may be logged or monitored without further notice, and that the resulting logs may be used as evidence in court.
- Specific notices required by specific local laws.

From a security, rather than a legal, point of view, your login banner usually should not contain any specific information about your router, its name, its model, what software it is running, or who owns it because this kind of information may be abused by an attacker.

For more information about the **banner login** command, see the following website:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/fun_r/cfr_1g01.htm#wp1029652

Role-Based CLI Access

Role-based CLI access is a feature introduced in Cisco IOS software Release 12.3(7)T that provides better control over access to Cisco IOS software devices. This feature allows the definition and deployment of CLI views, which restrict user access to CLI commands and configuration information.

Each CLI view defines a set of EXEC and configuration commands that are explicitly allowed or denied, and the view can be associated with specific users or groups. By implementing views, administrators can control the commands that are accepted and the configuration information that is visible to different groups of users.

For example, in many organizations the administration of the network resources is divided between security and network operations. In this situation, two CLI views can be defined. The network operations could have limited access to EXEC and configuration commands, and no access to security configuration commands. The security operations view would allow crypto EXEC commands and all security configuration commands.

In addition to user-configurable CLI views, role-based CLI access provides the following features:

- Root view
- Superviews
- Lawful intercept view

The root view is the highest administrative view, which has all access privileges, equivalent to a user with level 15 privileges. It is only from the root view that new CLI views can be defined or configured. To define a new view or add or remove commands from a view, the administrator must be in the root view.

A superview consists of a concatenation of multiple CLI views. This allows administrators to associate users or groups to a single superview rather than to multiple CLI views. Superviews do not include any commands, and in fact, commands can only be configured in CLI views. However, users logged into a superview can access all the commands that are configured for any of the CLI views that are part of the superview.

The lawful intercept view is available only in images that contain the lawful intercept subsystem. As the name indicates; this is a view that restricts access to lawful intercept commands and configuration information. Specifically, a lawful intercept view allows a user to secure access to lawful intercept commands that are held within the TAP-MIB, which is a special set of simple network management protocol (SNMP) commands that store information about calls and users.

By default , there is one root view. A total of 15 CLI views and superviews can be defined, and only one lawful intercept view can be created.

To define a new view, an enable password must exist, AAA must be enabled with the **aaa new-model** command, and the administrator must have level 15 privileges to access the root view. Additionally, to apply role-based CLI access to the console, the administrator must configure the **aaa authorization console** global command. By default, authorization is disabled for console access, and therefore role-based CLI access has no effect on the console. The **aaa authorization console** command overrides the default behavior and enables authorization on the console.

To access the root view, the administrator can use the **enable view** command, as shown in the following example:

```
Router# enable view
Password: (enter enable or enable secret password)
*Mar  18 00:04:28.891: %PARSER-6-VIEW_SWITCH: successfully set to view 'root'
Router#
```

Once in the root view, the administrator can create a new view by using the **parser view** command, as illustrated in the following example. A view password needs to be defined with the **password** command before additional attributes for the view can be configured.

```
Router# configure terminal
Router(config)# parser view Admin123
*Mar  18 01:07:56.167: %PARSER-6-VIEW_CREATED: view 'Admin123' successfully created.
Router(config-view)#
Router(config-view)# password 5 Admin@Pswd
```

After the CLI view is created, permitted or excluded commands can be added to the view with the **commands** command, as shown in the following example:

```
Router(config-view)# commands exec include show interfaces
Router(config-view)# commands exec include all
Router(config-view)# commands configure include-exclusive crypto
```

The following commands illustrate how to implement a network operations and security operations view, as described in the previous example. First, the following commands are required to configure the network operations view:

```
Router(config)# parser view NetOps
Router(config-view)# password 5 NetOps@Pswd
Router(config-view)# commands exec include clear
Router(config-view)# commands exec include copy
Router(config-view)# commands exec include ping
Router(config-view)# commands exec include all show
Router(config-view)# commands exec include configure
Router(config-view)# commands configure include access-list
Router(config-view)# commands configure include clock
Router(config-view)# commands configure include hostname
Router(config-view)# commands configure include interface
Router(config-view)# commands configure include ip
Router(config-view)# commands configure include line
Router(config-view)# exit
Router(config)#
```

The following commands configure the security operations view, which allows crypto EXEC commands and all security configuration commands:

```
Router(config)# parser view SecOps
Router(config-view)# password 5 SecOps@Pswd
Router(config-view)# commands exec include copy running-config
Router(config-view)# commands exec include login
Router(config-view)# commands exec include all show
Router(config-view)# commands exec include-exclusive show crypto
Router(config-view)# commands exec include-exclusive show key
Router(config-view)# commands exec include configure terminal
Router(config-view)# commands configure include access-list
Router(config-view)# commands configure include-exclusive crypto
Router(config-view)# commands configure include-exclusive key
Router(config-view)# commands configure include-exclusive li-view
Router(config-view)# exit
Router(config)#
```

Once a CLI view is defined, there are basically two ways a user or a group can be associated with the view: with a AAA local user database or with an external AAA server. Using an external AAA server is the preferred option. Both ways require that AAA is first enabled with the **aaa new-model** command.

Local users can be associated with a CLI view with the **username** command, as shown in the following example:

```
Router(config)# username admin_o view operator password chF&91$
Router(config)# username admin_n view NetOps password kz7pE%t
```

```
Router(config)# username admin_s view SecOps password p8eWo*i
```

This causes users to automatically enter the assigned view after a successful login.

Views can also be assigned by using an external AAA server using the new attribute **cli-view-name.** As mentioned previously, the best way to deploy authentication is by using AAA protocols like TACACS+, RADIUS or Kerberos.

For more information on role-based CLI access, including lawful intercept views,, see the following website:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps5207/products_feature_guide09186a00801ee18d.html#wp1076828

Appendix C—Commonly Used Protocols in the Infrastructure

Table 2 contains a list of commonly used protocols found in the infrastructure.

Table 2 Commonly Used Protocols in the Infrastructure

Protocol	Protocol Number, TCP/UDP Ports, Message Type
BGP	TCP/179
OSPF	Prot 89
EIGRP	Prot 88
GRE	Prot 47
AH	Prot 51
ESP	Prot 50
TACACS+	TCP/49
RADIUS	UDP/1812, UDP/1813; in the past UDP/1645 and UDP/1646
SSH	TCP/22
TELNET	TCP/23
SNMP	UDP/161
NTP	UDP/123
ICMP	Prot 1, ttl-exceeded, port-unreachable, echo, echo-reply
DNS	UDP/53