CHAPTER 4

# Branch Router QoS Design

This chapter discusses Branch QoS considerations and designs, including the following:

- AutoQoS—Enterprise
- Unidirectional Applications
- Branch LAN edge ingress classification
- Branch NBAR policies for worm identification and policing

Chapter 3, "WAN Aggregator QoS Design," discussed the QoS design recommendations for WAN aggregators in detail. For the most part, theses designs also apply to branch routers located at the far end of the WAN links. However, at least four unique considerations must be made for branch router QoS design. This chapter examines in detail these considerations and their related designs.

One of the first considerations is whether to configure the QoS policies manually or to utilize the Cisco Automatic QoS for the Enterprise feature. AutoQoS—Enterprise can automatically detect and provision bandwidth for up to 10 classes of traffic. This feature is well suited to smaller WAN/Branch networks managed by administrators with moderate QoS expertise. However for larger WAN/Branch networks, where centralized policies are generally preferred, this feature may not be appropriate.

If QoS policies are to be defined manually, then other considerations must also be taken into account, such as the presence of unidirectional applications. Some applications, such as Streaming-Video (whether unicast or multicast), require bandwidth allocation only on the WAN aggregator's WAN edge, not on the branch router's WAN edge. Therefore, bandwidth allocated to unidirectional applications on the WAN aggregator WAN edge can be redistributed among other preferential classes on the branch router's WAN edge.

Another characteristic common to branches is that traffic destined to the campus might not be correctly marked on the branch access switches. These switches, which are usually lower-end switches, might or might not have the capabilities to classify by Layer 3 or 4 parameters and mark DSCP values for data applications. Therefore, classification and marking might need to be performed on the branch router's LAN edge in the ingress direction. Furthermore, branch routers provide the capability to use NBAR to classify and mark flows that require stateful packet inspection.

Related to classification and NBAR, another unique consideration to branch QoS design is that branch routers are a strategic place to deploy NBAR policies for worm identification and policing. NBAR policies can be used to identify and drop Code Red, NIMDA, SQL Slammer, RPC DCOM/W32/MS Blaster, Sasser, and other worms.

Figure 4-1 shows the QoS policies required on a remote branch router.

*Figure 4-1 Branch Router QoS Policies*

# Branch WAN Edge QoS Design

WAN edge considerations discussed in Chapter 3, "WAN Aggregator QoS Design" apply also to the branch router's WAN edge. This includes the following:

- Link-speed categories (slow, medium, and high) and their respective design implications
- Bandwidth-provisioning guidelines (at least 25 percent for Best-Effort traffic and no more than 33 percent for Real-Time applications)
- Link-specific caveats (such as the fact that tools such as LFI and cRTP, when enabled, must be enabled on both ends of the WAN link for them to function correctly)

Distributed platform idiosyncrasies are not as relevant for branch router designs because these platforms rarely are deployed at remote sites (although they can be used for site-to-site links, as discussed in Chapter 3, "WAN Aggregator QoS Design").

## AutoQoS—Enterprise

For small-to-medium businesses supported by administrators with moderate QoS expertise, Cisco AutoQoS—Enterprise may be an expeditious option to enable QoS for WAN/Branch networks. This advanced IOS feature detects and automatically provisions bandwidth for up to 10 classes of traffic on a link-by-link basis based on an analysis performed on sampled traffic rates. Some considerations to keep in mind when deciding whether to use the AutoQoS—Enterprise feature:

- **Policies vary on a link-by-link basis**—AutoQoS—Enterprise samples traffic rates by application classes on a link-by-link basis and then presents a configuration recommendation based on these sampled rates. These rates vary link-by-link, so the recommended policies also vary link-by-link. AutoQoS— Enterprise is not a suitable tool for enterprises that want to centralize QoS policies for consistent end-to-end service-levels for their traffic classes.
- **Policies are reactive to traffic conditions during the sampling period, rather than administratively proactive**—The sampled traffic rates, on which the AutoQoS—Enterprise feature bases its recommendations, may or may not be inline with the desired business objectives of the QoS deployment. For example, AutoQoS—Enterprise may detect and provision bandwidth for a Streaming-Video class, but perhaps Streaming-Video is not viewed by the enterprise as a critical

application requiring explicit bandwidth guarantees; they might instead prefer to use the bandwidth to increase the service-levels to Transactional Data applications. In such a case, policies would have to be manually defined.

- **AutoQoS— Enterprise does not support a Mission-Critical Data class**—If an enterprise requires the support of a Mission-Critical Data class, which is a subjective evaluation of relative business priority of Transactional/Interactive applications (as discussed in Chapter 1, "Quality of Service Design Overview"), such a class would have to be manually defined.
- **AutoQoS—Enterprise can be used as a template**—The policies proposed by AutoQoS—Enterprise can be manually modified and tweaked to meet custom requirements. The configuration generated by AutoQoS—Enterprise is not an "all or nothing" option, but rather is modifiable and thus can be viewed as a template.
- **Incremental CPU load of NBAR**—Because AutoQoS—Enterprise relies heavily on NBAR—which generates a degree of incremental CPU load—we generally do not recommended that you enable it on large WAN networks, in particular on WAN Aggregation routers serving a large number of remote branches. AutoQoS— Enterprise is therefore more suitable for smaller WAN/Branch environments.
- **Link-specific restrictions**—AutoQoS— Enterprise has several link-specific restrictions, including:
  - **Serial Interface Restrictions**—For a serial interface with a low-speed link, MLP is configured automatically. The serial interface must have an IP address. When MLP is configured, this IP address is removed and put on the MLP bundle. To ensure that the traffic goes through the low-speed link, the following conditions must be met:
    - The AutoQoS for the Enterprise feature must be configured at both ends of the link.
    - The amount of bandwidth configured must be the same on both ends of the link.
  - **Frame Relay DLCI Restrictions**:
    - This feature cannot be configured on a Frame Relay DLCI if a map class is attached to the DLCI.
    - If a Frame Relay DLCI is already assigned to one subinterface, the AutoQoS for the Enterprise feature cannot be configured for a different subinterface.
    - For low-speed Frame Relay DLCIs configured for use on Frame Relay-to-ATM networks, MLP over Frame Relay (MLPoFR) is configured automatically. The subinterface must have an IP address. When MLPoFR is configured, this IP address is removed and put on the MLP bundle. The AutoQoS for the Enterprise feature must also be configured on the ATM side of the network.
    - For low-speed Frame Relay DLCIs with Frame Relay-to-ATM Interworking, the AutoQoS for the Enterprise feature cannot be configured if a virtual template is already configured for the DLCI.
  - **ATM PVC Restrictions**:
    - For a low-speed ATM PVC, the AutoQoS for the Enterprise feature cannot be configured if a virtual template is already configured for the ATM PVC.
    - For low-speed ATM PVCs, MLP over ATM (MLPoATM) is configured automatically. The subinterface must have an IP address. When MLPoATM is configured, this IP address is removed and put on the MLP bundle. The AutoQoS for the Enterprise feature must also be configured on the ATM side of the network.

The AutoQoS for the Enterprise feature consists of two configuration phases, completed in the following order:

1. **Auto-Discovery (data collection)**—The Auto-Discovery phase can be configured to operate in one of two modes:
   - **Untrusted Mode**—In untrusted mode (configured using the **auto discovery qos** interface command), the Auto-Discovery phase uses NBAR protocol discovery to detect the applications on the network and performs statistical analysis on the network traffic.
   - **Trusted Mode**—In trusted mode (configured using the **auto discovery qos trust** interface command), the Auto-Discovery phase classifies packets based on DSCP values in the IP header and collects the statistics to calculate bandwidth and average rate/peak rate and passes that data to the template module.
2. **AutoQoS Template Generation and Installation**—This phase (configured using the **auto qos** command) generates templates from the data collected during the Auto-Discovery phase and installs the templates on the interface. These templates are then used as the basis for creating the class maps and policy maps for your network. After the class maps and policy maps are created, they are then installed on the interface.

AutoQoS— Enterprise discovers and classifies as many as 10 classes of traffic, based on the QoS Baseline model. The only traffic class not automatically detected and classified is the Mission-Critical Data class, as this class requires a subjective evaluation on relative business priority vis-à-vis other Transactional/Interactive Data applications (as described in Chapter 1, "Quality of Service Design Overview").

These AutoQoS classes are described in Table 4-1.

*Table 4-1 AutoQoS for the Enterprise Feature Class Definitions*

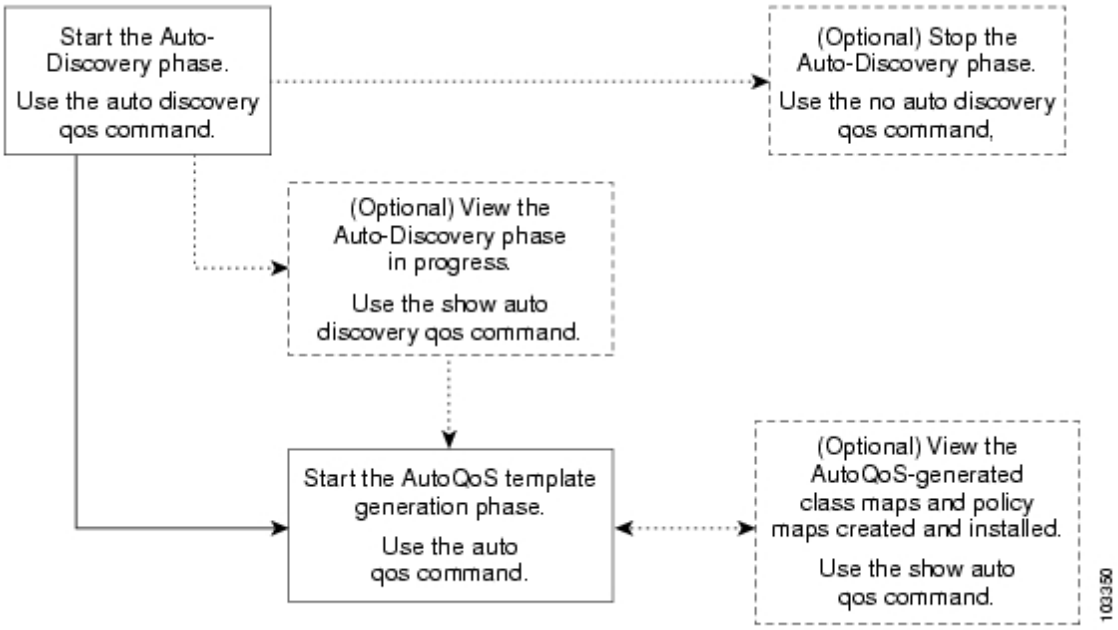
| AutoQoS Class Name | Traffic Type | DSCP Value |
|---|---|---|
| IP Routing | Network control traffic, such as routing protocols | CS6 |
| Interactive Voice | Inactive voice-bearer traffic | EF |
| Interactive Video | Interactive video data traffic | AF41 |
| Streaming Video | Streaming media traffic | CS4 |
| Telephony Signaling | Telephony signaling and control traffic | CS3 |
| Transactional/Interactive | Database applications transactional in nature | AF21 |
| Network Management | Network management traffic | CS2 |
| Bulk Data | Bulk data transfers; web traffic; general data service | AF11 |
| Scavenger | Casual entertainment; rogue traffic; traffic in this category is given less-than-best-effort treatment | CS1 |
| Best Effort | Default class; all non-critical traffic; HTTP; all miscellaneous traffic | 0 |

Additionally, the AutoQoS—Enterprise feature includes the following verification commands:

- **show auto discovery qos**
- **show auto qos**

The top-level processes for configuring and verifying the AutoQoS—Enterprise feature are illustrated in Figure 4-2.

*Figure 4-2 Top-Level Processes for Configuring the AutoQoS for the Enterprise Feature*




As noted, the AutoQoS— Enterprise feature is a handy alternative for automating QoS deployment in smaller WAN/Branch environments. For more information, see the IOS documentation:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123t/123t_11/ft_aqose.htm

If, on the other hand, all or some of the branch QoS policies must be defined manually, then additional considerations must be taken into account, as discussed below.

# Unidirectional Applications

Some applications are completely symmetrical and require identical bandwidth provisioning on both ends of the WAN link. For example, if 100 kbps of LLQ are assigned to voice in one direction, 100 kbps of LLQ also must be provisioned for voice in the opposite direction (assuming that the same VoIP codecs are being used in both directions, and putting aside for a moment multicast Music-on-Hold [MoH] provisioning). Furthermore, having symmetrical polices on both sides of the WAN links greatly simplifies QoS policy deployment and management, which is an important aspect of large-scale designs.

However, certain applications, such as Streaming-Video and multicast MoH, most often are unidirectional. Therefore, it might be unnecessary and even inefficient to provision any bandwidth guarantees for such traffic on the branch router for the branch-to-campus direction of traffic flow.

Most applications lie somewhere in the middle of the scale, between the extremes of being fully bidirectional and being completely unidirectional. Most client/server applications lie closer to the unidirectional end of the scale because these applications usually consist of small amounts of client-to-server traffic coupled with larger amounts of server-to-client traffic. Such behavior can be reflected in the asymmetrical bandwidth provisioning for such types of applications.

For purely unidirectional applications, it is recommended that provisioning be removed from the WAN edge policies on branch routers and the allocated bandwidth be redistributed among other classes.

## Branch Router WAN Edge (10-Class) QoS Baseline Model

The inclusion or exclusion of the Streaming-Video class affects only those enterprises deploying complex QoS class models, such as the 11-Class QoS Baseline Model. When the Streaming-Video class is removed from this model (for branch router WAN edges), the bandwidth previously allocated to this class can be reallocated among the other preferential data classes, as illustrated in Figure 4-3. Notice that no class is provisioned for Streaming-Video in this model, and the bandwidth assigned to it from the 11-Class WAN edge model (from Figure 3-6 in the previous chapter) has been reassigned to the Mission-Critical Data class and the Transactional Data class.

The configuration for such a branch router (10-class) QoS Baseline policy on a dual-T1 interface is shown in Example 4-1. Notice that there is no Streaming-Video class and that the bandwidth for it has been distributed equally between the Mission-Critical Data class (now at 15 percent instead of 10 percent) and the Transactional Data class (now at 12 percent instead of 7 percent).

A **bandwidth** statement is used on class-default, precluding the use of **fair-queue** on the class for all nondistributed platforms. Also, a **max-reserved-bandwidth 100** command must be issued on the interface before the **service-policy output** command.

*Figure 4-3 Branch Router (10-Class) QoS Baseline WAN Edge Model Bandwidth Allocations Example (Dual-T1 Link Example)*

*Example 4-1 Branch Router (10-Class) QoS Baseline WAN Edge Model*

```
!
class-map match-all VOICE
  match ip dscp ef            ! IP Phones mark Voice to EF
class-map match-all INTERACTIVE-VIDEO
  match ip dscp af41 af42     ! Recommended markings for IP/VC
class-map match-any CALL-SIGNALING
  match ip dscp cs3           ! Future Call-Signaling marking
```

```
  match ip dscp af31          ! Current Call-Signaling marking
class-map match-all ROUTING
  match ip dscp cs6           ! Routers mark Routing traffic to CS6
class-map match-all NET-MGMT
  match ip dscp cs2           ! Recommended marking for Network Management
class-map match-all MISSION-CRITICAL-DATA
  match ip dscp 25            ! Interim marking for Mission-Critical Data
class-map match-all TRANSACTIONAL-DATA
  match ip dscp af21 af22     ! Recommended markings for Transactional Data
class-map match-all BULK-DATA
  match ip dscp af11 af12     ! Recommended markings for Bulk Data
class-map match-all SCAVENGER
  match ip dscp cs1           ! Recommended marking for Scavenger traffic
!
 policy-map BRANCH-WAN-EDGE
  class VOICE
   priority percent 18        ! Voice gets 552 kbps of LLQ
  class INTERACTIVE-VIDEO
   priority percent 15        ! 384 kbps IP/VC needs 460 kbps of LLQ
  class CALL-SIGNALING
   bandwidth percent 5        ! Minimal BW guarantee for Call-Signaling
  class ROUTING
   bandwidth percent 3        ! Routing class gets 3% explicit BW guarantee
  class NET-MGMT
   bandwidth percent 2        ! Net-Mgmt class gets 2% explicit BW guarantee
  class MISSION-CRITICAL-DATA
   bandwidth percent 15       ! Mission-Critical class gets min 15% BW guarantee
   random-detect              ! Enables WRED on Mission-Critical Data class
  class TRANSACTIONAL-DATA
   bandwidth percent 12       ! Transactional-Data class gets min 12% BW guarantee
   random-detect dscp-based   ! Enables DSCP-WRED on Transactional-Data class
  class BULK-DATA
   bandwidth percent 4        ! Bulk Data class gets 4% BW guarantee
   random-detect dscp-based   ! Enables DSCP-WRED on Bulk-Data class
  class SCAVENGER
   bandwidth percent 1        ! Scavenger class is throttled
  class class-default
   bandwidth percent 25       ! Default class gets min 25% BW guarantee
   random-detect              ! Enables WRED on the default class
!
```

Verification commands:

- **show policy**
- **show policy interface**

Now that considerations and designs for the WAN edge of the branch router have been addressed, the next section discusses the LAN edge.

# Branch Router LAN Edge QoS Design

The LAN edge of the branch router can have egress and ingress policies. Because you have been dealing with egress policies since the WAN/branch discussion began, and because the egress policies are not only optional, but also considerably simpler, they are discussed first.

As previously mentioned, it is better to mark at Layer 3 (DSCP) instead of Layer 2 whenever possible because Layer 2 markings are lost when the transmission medium changes. This is the case with any Ethernet 802.1Q/p CoS values that have been set within the campus and are carried over a WAN (or VPN). In some cases, network administrators prefer to have these markings restored at the branch; DSCP-to-CoS mapping then can be performed on the branch router's LAN edge.

In the ingress direction, the branch router might be required to perform classification and marking of branch-to-campus traffic. This might be because the branch switch lacks the capability to classify and mark traffic, or because the traffic consists of stateful flows that require NBAR for classification. NBAR classification also is required at branch LAN ingress edges to identify (and immediately drop) known worm traffic.

Each of these types of polices is discussed in more detail in the following sections.

# DSCP-to-CoS Remapping

DSCP-to-CoS remapping is optional. Newer Catalyst switches perform QoS based on internal DSCP values that are generated either by trusted DSCP markings or by trusted CoS markings (coupled with CoS-to-DSCP mappings). In the case of legacy switches at the branch that perform QoS strictly by preset CoS values, CoS might need to be remapped on the branch router's LAN edge.

Enhanced packet marking (Cisco IOS Release 12.2[13]T or higher) is the optimal tool for resetting CoS values because it uses a table. In this manner, a default DSCP-to-CoS mapping can be used without having to configure explicitly a class-based marking policy that matches every DiffServ class and performs a corresponding **set cos** function.

In both cases, keep in mind that only Ethernet trunking protocols, such as 802.1Q, carry CoS information. Therefore, the policy works correctly only when applied to a trunked subinterface, not to a main Ethernet interface.

Example 4-2 presents an enhanced packet marking DSCP-to-CoS configuration for a branch LAN edge. Note that this DSCP-to-CoS remapping policy requires application to both the voice VLAN (VVLAN) subinterface and the data VLAN (DVLAN) subinterface on the branch router's LAN edge.

***Example 4-2 Branch LAN Edge Enhanced Packet Marking for DSCP-to-CoS Remapping Example***

```
!
ip cef                                  ! IP CEF is Required for Packet Marking
!
policy-map BRANCH-LAN-EDGE-OUT
  class class-default
   set cos dscp                         ! Enables default DSCP-to-CoS Mapping
!
!
interface FastEthernet0/0
 no ip address
 speed auto
 duplex auto
!
interface FastEthernet0/0.60
 description DVLAN SUBNET 10.1.60.0
 encapsulation dot1Q 60
 ip address 10.1.60.1 255.255.255.0
 service-policy output BRANCH-LAN-EDGE-OUT       ! Restores CoS for Data VLAN
!
interface FastEthernet0/0.160
 description VVLAN SUBNET 10.1.160.0
 encapsulation dot1Q 160
```

```
 ip address 10.1.160.1 255.255.255.0
 service-policy output BRANCH-LAN-EDGE-OUT        ! Restores CoS on Voice VLAN
!
```

Verification commands:

- **show policy**
- **show policy interface**

# Branch-to-Campus Classification and Marking

In keeping with the unofficial Differentiated Services design principle of marking traffic as close to its source as possible, IP phones mark voice-bearer traffic (to DSCP EF) and Call-Signaling traffic (currently, to DSCP AF31, but this will soon change to DSCP CS3) on the phones themselves. Some IP/VC devices mark Interactive-Video traffic to AF41 on their network interface cards (NICs).

However, as has already been discussed, it is generally not recommended that end-user PCs be trusted to set their CoS/DSCP markings correctly because users easily can abuse this (either unintentionally or deliberately).

**Note** There may be some exceptions to this general rule, such as PCs running applications like Cisco VT Advantage in controlled environments. A network administrator must make these policy decisions.

Therefore, application traffic that originates from untrusted hosts should be marked on branch access switches. However, in some circumstances, this might not be possible:

- The branch access switch does not have Layer 3 or Layer 4 awareness for traffic classification or does not support marking.
- Classification needs to be performed at the application layer (through NBAR).

In such cases, DSCP classification must be performed at the ingress interface of the branch router.

Administrators can identify and then mark application traffic based on the following criteria:

- **Source or destination IP address (or subnet)**—Typically, destination subnets are used when setting branch QoS policies. For example, the destination subnet for a group of application servers can be used to identify a particular type of client-to-server application traffic.
- **Well-known TCP/UDP ports**—It is important to know whether the well-known port is a source port or a destination port from the branch router's perspective.
- **NBAR protocol (for example, Citrix or KaZaa) or application subparameter (for example, HTTP URL)**—NBAR Packet Description Language Modules (PDLMs) also can be configured to identify stateful or proprietary applications, as well as worms.

Regardless of the method used to identify the application traffic, the inbound classification and marking policy needs to be applied only to the DVLAN subinterface. This is because only trusted IP Telephony applications, which mark their voice traffic correctly, are admitted onto the voice VLAN.

**Note** If the branch access switches support access-edge policers (as described in Chapter 2, "Campus QoS Design"), these likewise should be enabled on them. This adds another layer of defense to the network, mitigating DoS/worm traffic that originates from the branch through Scavenger-class QoS.

If the branch access switches do not support such policing, compatible policers could be placed at the branch router access edge. This is in harmony with the principle of policing as close to the source as

possible.

Whenever possible, though, such policing should be done in Catalyst hardware rather than Cisco IOS Software.

Although it might be tempting to use the same class-map names (such as MISSION-CRITICAL, TRANSACTIONAL-DATA, or BULK-DATA) for ingress LAN edge classification and marking policies as are in use for egress WAN edge queuing policies, this might cause confusion in policy definition and troubleshooting. Therefore, for management and troubleshooting simplicity, it is beneficial to have similar yet descriptive names for these new classes (for example, branch-originated traffic might have class-map names prepended with "BRANCH-"). A description can also be added to the class maps with the **description** command.

## Source or Destination IP Address Classification

Example 4-3 shows how traffic destined to a specific subnet (10.200.200.0/24)—which, in this case, represents a server farm of proprietary Mission-Critical Data application servers—can be identified and marked on ingress on the branch router's LAN edge.

***Example 4-3 Branch LAN Edge Destination IP Classification and Marking Example***

```
!
ip cef                                                ! Required for Packet Marking
!
class-map match-all BRANCH-MISSION-CRITICAL
  match access-group name MISSION-CRITICAL-SERVERS    ! ACL to reference
!
 policy-map BRANCH-LAN-EDGE-IN
  class BRANCH-MISSION-CRITICAL
   set ip dscp 25   ! (Interim) Recommended marking for Mission-Critical traffic
!
...
!
interface FastEthernet0/0
 no ip address
 speed auto
 duplex auto
!
interface FastEthernet0/0.60
 description DVLAN SUBNET 10.1.60.0
 encapsulation dot1Q 60
 ip address 10.1.60.1 255.255.255.0
 service-policy output BRANCH-LAN-EDGE-OUT  ! Restores CoS on Data VLAN
 service-policy input BRANCH-LAN-EDGE-IN    ! Marks MC Data on ingress
!
...
!
ip access-list extended MISSION-CRITICAL-SERVERS
 permit ip any 10.200.200.0 0.0.0.255    ! MC Data Server-Farm Subnet
!
```

Verification commands:

- **show policy**
- **show policy interface**
- **show ip access-list**

## Verification Command: show ip access-list

When access lists are used as the filtering criteria for a class map, the **show ip access-list** command is helpful in identifying whether the access list is registering matches, especially for ACLs that have multiple lines of match criteria. This command provides granular visibility into which lines of an access list are registering matches. In Example 4-4, 464 matches are registered against the access list.

***Example 4-4 show ip access-list Verification of Remote Branch LAN Edge Destination IP Classification Example***

```
BRANCH#60-C3745#show ip access-list MISSION-CRITICAL-SERVERS
Extended IP access list MISSION-CRITICAL-SERVERS
    10 permit ip any 10.200.200.0 0.0.0.255 (464 matches)
BRANCH#60-C3745#
```

## Well-Known TCP/UDP Port Classification

Most applications can be identified by their well-known TCP/UDP ports. Some of these ports have keywords within Cisco IOS Software to identify them when defining access lists.

**Note** The Internet Assigned Numbers Authority (IANA) lists registered well-known and registered application ports at http://www.iana.org/assignments/port-numbers.

Building on Example 4-4, Example 4-5 classifies and marks branch-originated FTP and e-mail traffic, both POP3 and IMAP (TCP port 143), as Bulk Data (DSCP AF11).

***Example 4-5 Branch LAN Edge Well-Known Port Classification Example***

```
!
ip cef                                          ! Required for Packet Marking
!
class-map match-all BRANCH-MISSION-CRITICAL
  match access-group name MISSION-CRITICAL-SERVERS
class-map match-all BRANCH-BULK-DATA
  match access-group name BULK-DATA-APPS      ! ACL to reference
!
 policy-map BRANCH-LAN-EDGE-IN
  class BRANCH-MISSION-CRITICAL
   set ip dscp 25
  class BRANCH-BULK-DATA
   set ip dscp af11                           ! Bulk data apps are marked to AF11
!
!
interface FastEthernet0/0
 no ip address
 speed auto
 duplex auto
!
interface FastEthernet0/0.60
 description DVLAN SUBNET 10.1.60.0
 encapsulation dot1Q 60
 ip address 10.1.60.1 255.255.255.0
 service-policy output BRANCH-LAN-EDGE-OUT      ! Restores CoS on Data VLAN
 service-policy input BRANCH-LAN-EDGE-IN        ! Marks Data on ingress
!
...
```

```
!
ip access-list extended MISSION-CRITICAL-SERVERS
 permit ip any 10.200.200.0 0.0.0.255
!
ip access-list extended BULK-DATA-APPS
 permit tcp any any eq ftp                    ! Identifies FTP Control traffic
 permit tcp any any eq ftp-data               ! Identifies FTP Data traffic
 permit tcp any any eq pop3                   ! Identifies POP3 E-mail traffic
 permit tcp any any eq 143                    ! Identifies IMAP E-mail traffic
!
```

Verification commands:

- **show policy**
- **show policy interface**
- **show ip access-list**

## NBAR Application Classification

At the time of this writing, Cisco IOS Software included NBAR PDLMs for 98 of the most common network applications, with the capability to define an additional 10 applications using custom PDLMs.

Of these protocols, 15 require stateful packet inspection for positive identification. Because NBAR operates in the IP Cisco Express Forwarding (CEF) switching path, only the first packet within a flow requires stateful packet inspection, and the policy is applied to all packets belonging to the flow. NBAR stateful packet inspection requires more CPU processing power than simple access control lists (ACLs). However, on newer branch router platforms, such as Cisco's 1800, 2800, and 3800 series Integrated Services Routers (ISRs), Cisco Technical Marketing testing has shown the overhead of enabling NBAR classification at dual-T1 rates to be quite minimal (typically 2 to 5 percent, depending on the traffic mix).

Building again on the previous example, Example 4-6 uses NBAR to identify several types of Transactional Data applications, Network-Management applications, and Scavenger applications.

In Example 4-6, Transactional Data applications include Citrix, LDAP, Oracle SQL*NET, and HTTP web traffic with "SalesReport" in the URL. In addition, a custom PDLM is defined to identify SAP traffic (by TCP ports 3200 through 3203 and 3600), and SAP also is included as a Transactional Data application.

Additionally, Network-Management applications, such as SNMP, Syslog, Telnet, NFS, DNS, ICMP, and TFTP, are identified through NBAR and marked to DSCP CS2.

Furthermore, Scavenger applications, such as Napster, Gnutella, KaZaa (versions 1 and 2), Morpheus, Grokster, and many other peer-to-peer file-sharing applications, are identified by NBAR and marked to DSCP CS1.

***Example 4-6 Branch LAN Edge NBAR Classification Example***

```
!
ip nbar port-map custom-01 tcp 3200 3201 3202 3203 3600   ! PDLM Mapping for SAP
!
ip cef     ! IP CEF is required for both Class-Based Marking and for NBAR
!
class-map match-all BRANCH-MISSION-CRITICAL
  match access-group name MISSION-CRITICAL-SERVERS
class-map match-any BRANCH-TRANSACTIONAL-DATA! Must use "match-any"
  match protocol citrix                       ! Identifies Citrix traffic
  match protocol ldap                        ! Identifies LDAP traffic
  match protocol sqlnet                      ! Identifies Oracle SQL*NET traffic
  match protocol http url "*SalesReport*"    ! Identifies "SalesReport" URLs
```

```
  match protocol custom-01        ! Identifies SAP traffic via Custom-01 PDLM Port-Map
class-map match-all BRANCH-BULK-DATA
  match access-group name BULK-DATA-APPS
class-map match-any BRANCH-NET-MGMT
  match protocol snmp                          ! Identifies SNMP traffic
  match protocol syslog                        ! Identifies Syslog traffic
  match protocol telnet                        ! Identifies Telnet traffic
  match protocol nfs                           ! Identifies NFS traffic
  match protocol dns                           ! Identifies DNS traffic
  match protocol icmp                          ! Identifies ICMP traffic
  match protocol tftp                          ! Identifies TFTP traffic
class-map match-any BRANCH-SCAVENGER
  match protocol napster                       ! Identifies Napster traffic
  match protocol gnutella                      ! Identifies Gnutella traffic
  match protocol fasttrack                     ! Identifies KaZaa (v1) traffic
  match protocol kazaa2                        ! Identifies KaZaa (v2) traffic
!
policy-map BRANCH-LAN-EDGE-IN
  class BRANCH-MISSION-CRITICAL
   set ip dscp 25
  class BRANCH-TRANSACTIONAL-DATA
   set ip dscp af21           ! Transactional Data apps are marked to DSCP AF21
  class BRANCH-NET-MGMT
   set ip dscp cs2            ! Network Management apps are marked to DSCP CS2
  class BRANCH-BULK-DATA
   set ip dscp af11
  class BRANCH-SCAVENGER
   set ip dscp cs1             ! Scavenger apps are marked to DSCP CS1
!
...
!
interface FastEthernet0/0
 no ip address
 speed auto
 duplex auto
!
interface FastEthernet0/0.60
 description DVLAN SUBNET 10.1.60.0
 encapsulation dot1Q 60
 ip address 10.1.60.1 255.255.255.0
 service-policy output BRANCH-LAN-EDGE-OUT  ! Restores CoS on Data VLAN
 service-policy input BRANCH-LAN-EDGE-IN    ! Input Marking policy on DVLAN only
!
...
!
ip access-list extended MISSION-CRITICAL-SERVERS
 permit ip any 10.200.200.0 0.0.0.255
!
!
ip access-list extended BULK-DATA-APPS
 permit tcp any any eq ftp
 permit tcp any any eq ftp-data
 permit tcp any any eq pop3
 permit tcp any any eq 143
!
```

**Note** The NBAR **fasttrack** PDLM identifies KaZaa (version 1), Morpheus, Grokster, and other applications. The NBAR **gnutella** PDLM identifies Gnutella, BearShare, LimeWire, and other peer-to-peer applications.

Verification commands:

- **show policy**
- **show policy interface**
- **show ip access-list**
- **show ip nbar port-map**

## Verification Command: show ip nbar port-map

When NBAR custom PDLMs are used to identify applications, it is useful to verify that the ports bound to the PDLM have been entered correctly. This information is obtained with the **show ip nbar port-map** command. In this (filtered) example, Example 4-7, the TCP ports given for SAP (3200 through 3203 and 3600) were bound correctly to the Custom-01 NBAR PDLM.

***Example 4-7*** **show ip nbar port-map** ***Verification Custom-PDLM TCP/UDP Port-Mapping Example***

```
BRANCH#60-C3745#show ip nbar port-map | include custom-01
port-map custom-01            tcp 3200 3201 3202 3203 3600
BRANCH#60-C3745#
```

# NBAR Known-Worm Classification and Policing

Worms are nothing new. They've been around almost as long as the Internet itself (one of the first Internet worms was the Morris worm, released in November 1988). Typically, worms are self-contained programs that attack a system and try to exploit a vulnerability in the target. Upon successfully exploiting the vulnerability, the worm copies its program from the attacking host to the newly exploited system to begin the cycle again.

**Note** A virus, which is slightly different from a worm, requires a vector to carry the virus code from one system to another. The vector can be either a word-processing document, an e-mail, or an executable program.

The main element that distinguishes a worm from a virus is that a computer virus requires human intervention to facilitate its spreading, whereas worms (once released) propagate without requiring additional human intervention.

Worms are comprised of three primary components (as illustrated in Figure 4-4):

- **The enabling exploit code**—The enabling exploit code is used to exploit a vulnerability on a system. Exploitation of this vulnerability provides access to the system and the capability to execute commands on the target system.
- **A propagation mechanism**—When access has been obtained through the enabling exploit, the propagation mechanism is used to replicate the worm to the new target. The method used to replicate the worm can be achieved through the use of the Trivial File Transfer Protocol (TFTP), FTP, or another communication method. When the worm code is brought to the new host, the cycle of infection can be started again.
- **A payload**—Some worms also contain payloads, which might include additional code to further exploit the host, modify data on the host, or change a web page. A payload is not a required component, and, in many cases, the worm's enabling exploit code itself can be considered the payload.

*Figure 4-4 Anatomy of an Internet Worm*

Some worms use unique TCP/UDP ports to propagate. These types of worms are fairly simple to block using access lists (when the ports are known). Such ACLs can be configured on the branch switch (whenever supported) or on the branch router's LAN edge.

Other worms hijack legitimate TCP/UDP ports to carry their harmful payloads. For these latter types of worms, NBAR can be used at the branch LAN edge to perform deep-packet analysis and drop any packets that are carrying the payloads of known worms. Some of these known worms include Code Red, NIMDA, SQL Slammer, RPC DCOM/W32/MS Blaster, and Sasser. The following sections discuss how NBAR can be used for each of these types of worms.

## NBAR Versus Code Red

First released in July 2001, Code Red targeted Microsoft Internet Information Server (IIS) using a vulnerability in the IIS Indexing Service. Although the first variant of this worm did little damage because of a flaw in the random number–generator code used to generate addresses of hosts to exploit, a second variant appeared with the flaw fixed.

This worm, CodeRedv2, spread quickly and became the most widespread and damaging worm to hit the Internet since the Morris worm. CodeRedv2's success as a worm relied on the fact that the worm exploited the vulnerability in the IIS Indexing Service only as a means of gaining access to the host. This, coupled with the wide deployment of IIS as well as the large number of unpatched IIS web servers, contributed to the quick and wide-ranging spread of the worm. An estimated 360,000 hosts were infected within a period of 14 hours.

CodeRedv2 temporarily replaced the home page of the web servers that it struck with a new page. Additionally, the code of the worm indicated that it was programmed to begin a packet-flooding DoS attack against a hard-coded IP address. (At the time, this was the IP address of the White House web server at http://www.whitehouse.gov.)

The original Code Red payload is shown in Example 4-8. The initial infection attempt sends a large HTTP GET request to the target IIS server.

*Example 4-8 Original Code Red Payload*

```
2001-08-04 16:32:23 24.101.17.216 - 10.1.1.75 80 GET /default.ida
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

```
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd
3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u
531b%u53ff%u0078%u0000%u00=a 403
```

The CodeRedv2 payload is shown in Example 4-9.

***Example 4-9 CodeRedv2 Payload***

```
2001-08-04 15:57:35 64.7.35.92 - 10.1.1.75 80 GET /default.ida
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u7801%u
9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u
8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a 403 -
```

Notice that the GET request in both cases is looking for a file named default.ida. However, this filename changes in newer variants of Code Red, such as CodeRedv3/CodeRed.C, as shown in Example 4-10.

***Example 4-10 CodeRedv3 Payload***

```
2001-08-06 22:24:02 24.30.203.202 - 10.1.1.9 80 GET /x.ida
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA=X 403 HTTP/1.1 -
```

Although the filename has changed, the .ida suffix remains the same.

Code Red variants can include payloads that execute cmd.exe or root.exe functions to program scripts within the IIS scripts directory, thus providing a ready-made back door to the server for any attacker to use.

Therefore, to combat Code Red, NBAR policies can be configured to check the payload of HTTP packets for these criteria (.ida, cmd.exe, and root.exe), as shown in Example 4-11.

***Example 4-11 NBAR Policies to Identify Code Red***

```
!
class-map match-any CODE-RED
   match protocol http url "*.ida*"          ! Identifies HTTP GET .ida requests
   match protocol http url "*cmd.exe*"       ! Identifies HTTP with cmd.exe
   match protocol http url "*root.exe*"      ! Identifies HTTP with root.exe
!
```

Verification command:

- show policy

## NBAR Versus NIMDA

Two months after Code Red struck the Internet, another large-scale worm, NIMDA, was released. Unlike Code Red, NIMDA was a hybrid worm because it contained the characteristics of both a worm and a virus. NIMDA spread using several vectors:

- Through e-mail as an attachment (virus vector)
- Through network shares (worm vector)

- Through JavaScript by browsing compromised websites (virus vector)
- Through infected hosts actively scanning for additional exploitable hosts (worm vector)
- Through infected hosts actively scanning for back doors created by Code Red (worm vector)

NIMDA did not appear to exhibit intentional destructive capabilities; to date, NIMDA's activities have been restricted to its self-propagation, which has the side effect of a DoS (flooding) attack.

NIMDA propagates itself by copying, downloading, or executing a file called readme.eml. Therefore, NBAR can be used to check the payload of HTTP packets to see if they are propagating this file, as shown in Example 4-12.

***Example 4-12 NBAR Policies to Identify NIMDA***

```
!
class-map match-any NIMDA
   match protocol http url "*readme.eml*"     ! Identifies HTTP with "readme.eml"
!
```

Verification command:

- **show policy**

## NBAR Versus SQL Slammer

After the NIMDA infection subsided, the Internet saw the appearance of smaller infectious worms. In January 2003, a new worm infected the Internet at such a high rate that it was categorized as a flash worm. This worm, termed SQL Slammer, once again targeted Microsoft Windows servers; specifically, this worm targeted servers running Microsoft Structured Query Language (SQL) Server software. The vulnerability exploited by SQL Slammer had been published in July 2002, and a patch from Microsoft was available at that time as well. Even though this patch was available for almost six months, SQL Slammer spread with incredibly high efficiency.

SQL slammer is a 376-byte User Datagram Protocol (UDP)–based worm that infects Microsoft SQL servers through UDP port 1434. Example 4-13 shows a signature string from the SQL Slammer worm.

***Example 4-13 SQL Slammer Worm Signature String***

```
\x04\x01\x01\x01\x01\x01.*[.][Dd][Ll][Ll]
```

Because of its small size, the SQL Slammer worm is contained in a single packet. The fast scanning rate of SQL Slammer is achieved not only because of this small size, but also because the worm is UDP based. The worm does not have to complete a handshake (necessary with TCP-based worms) to connect with a target system.

SQL Slammer reached its full scanning rate of 55 million scans per second within 3 minutes of the start of the infection and infected the majority of vulnerable hosts on the Internet within 10 minutes of the start of the infection, with an estimated 300,000 infected hosts overall. A major consequence of such a fast scanning rate was that edge networks were overwhelmed by the amount of traffic generated by the worm. SQL Slammer's doubling rate was approximately 8.5 seconds. In contrast, CodeRedv2's doubling rate was about 37 minutes.

SQL Slammer does not carry an additional harmful payload (beyond its enabling exploit code), and its primary purpose is to cause DoS through exponential self propagation.

NBAR can be used to detect the SQL Slammer worm by mapping a custom PDLM to UDP port 1434 and matching on the packet length (376-byte worm + 8 bytes of UDP header + 20 bytes of IP header = 404 bytes). This is shown in Example 4-14.

*Example 4-14 NBAR Policies to Identify SQL Slammer*

```
!
ip nbar port-map custom-02 udp 1434          ! Maps a custom PDLM to UDP 1434
!
class-map match-all SQL-SLAMMER
   match protocol custom-02                  ! Matches the custom Slammer PDLM
   match packet length min 404 max 404       ! Matches the packet length (376+28)
!
```

Verification commands:

- show policy
- show ip nbar port-map

**Note** Because NBAR custom-01 PDLM has been used in previous examples to identify SAP traffic, another custom PDLM (custom-02) is used in this example. Subsequent examples similarly are defined with the next-available custom PDLM.

## NBAR Versus RPC DCOM/W32/MS Blaster

First released in August 2003, the Remote Procedure Call (RPC) Distributed Component Object Model (DCOM) worm exploited a flaw in a section of Microsoft's RPC code dealing with message exchange over TCP/IP, resulting in the incorrect handling of malformed messages. This flaw was a stack-based buffer overflow occurring in a low-level DCOM interface within the RPC process listening on TCP ports 135, 139, and 445.

The DCOM protocol enables Microsoft software components to communicate with one another. This is a core function of the Windows kernel and cannot be disabled. The vulnerability results because the Windows RPC service does not properly check message inputs under certain circumstances. By sending a malformed RPC message, an attacker can cause the RPC service on a device to fail in such a way that arbitrary code could be executed. The typical exploit for this vulnerability launches a reverse-telnet back to the attacker's host to gain complete access to the target.

Successful exploitation of this vulnerability enables an attacker to run code with local system privileges. This enables an attacker to install programs; view, change, or delete data; and create new accounts with full privileges. Because RPC is active by default on all versions of the Windows operating system, any user who can deliver a malformed TCP request to an RPC interface of a vulnerable computer could attempt to exploit the vulnerability. It is even possible to trigger this vulnerability through other means, such as logging into an affected system and exploiting the vulnerable component locally.

A variant of the RPC DCOM worm is termed W32 Blaster or MS Blaster. When MS Blaster successfully exploits a host, it attempts to upload a copy of the worm program to the newly exploited host. MS Blaster uses TFTP to copy the worm program from the attacking host to the target system. MS Blaster also starts up a cmd.exe process and binds it to TCP port 4444 of the newly exploited system. This provides any attacker with direct command-line access at the local system privilege level, as discussed previously.

To access the system, the attacker needs only to telnet to TCP port 4444 on the exploited host. If the worm is successful in copying the MS Blaster program to the target, the worm exploit code modifies the system registry to ensure that the worm is restarted if the system reboots. It then launches the worm program on the newly exploited host to begin the cycle again, starting with scanning for more exploitable hosts. MS Blaster also contained code for a DoS attack. This particular attack was targeted at www.windowsupdate.com.

NBAR can be used to combat the RPC DCOM/W32/MS Blaster worm by identifying communications on TCP/UDP ports 135, 139, and 445.

By default, the NBAR **exchange** PDLM is mapped to TCP port 135; therefore, this PDLM can be used as part of the MS Blaster worm policy definition. Similarly, the NBAR **netbios** PDLM is bound by default to TCP/UDP port 139 (in addition to TCP port 137 and UDP ports 137 and 138), so this PDLM also can be used within the policy definition; specifically, the **netbios** PDLM can have its port mapping expanded to include TCP port 445 and UDP ports 135, 139, and 445, as shown in Example 4-15.

**Note** Alternatively, a custom PDLM can be defined for these ports (TCP/UDP 135, 139, and 445), but before this could be done, you would have to map the **exchange** and **netbios** PDLMs ports away from their defaults, to avoid conflicting PDLM port mappings.

***Example 4-15 NBAR Policies to Identify RPC DCOM/W32/MS Blaster***

```
!
ip nbar port-map netbios tcp 137 139 445             ! Matches TCP 137/139/445
ip nbar port-map netbios udp 135 137 138 139 445     ! Matches UDP 135/137-139/445
!
class-map match-any MS-BLASTER
   match protocol exchange                    ! Matches TCP port 135
   match protocol netbios                     ! Matches MS Blaster NetBIOS PDLM
!
```

Verification commands:

- **show policy**
- **show ip nbar port-map**

## NBAR Versus Sasser

The next major worm after MS Blaster was the Sasser worm (and variants Sasser.A/B/C/D), which was released in late April 2004. Sasser exploits a flaw in the Windows Local Security Authority Service Server (LSASS) that can cause systems to crash and continually reboot, or allow a remote attacker to execute arbitrary code with local system privileges.

Sasser is very efficient in scanning: It can scan 1024 separate IP addresses simultaneously (on TCP port 445). When scanning reveals a vulnerable system, the worm exploits the LSASS vulnerability and creates a remote shell (RSH) session on TCP port 9996 back to the infecting system. Then Sasser starts an FTP server on TCP port 5554 to retrieve a copy of the worm.

Sasser can be identified through a custom NBAR PDLM listening for communication on TCP ports 445, 5554, and 9996, as shown in Example 4-16.

**Note** If TCP port 445 already has been bound to the **netbios** NBAR PDLM (as recommended previously in the MS Blaster worm definition), it is not necessary to include this port in the Sasser custom PDLM port mapping (because it will cause a conflict).

***Example 4-16 NBAR Policies to Identify Sasser***

```
!
ip nbar port-map custom-03 tcp 445 5554 9996  ! Matches on TCP 445/5554/9996
!
class-map match-all SASSER
   match protocol custom-03                   ! Matches Sasser custom PDLM
!
```

Verification commands:

- show policy
- show ip nbar port-map

## NBAR Versus Future Worms

There is every reason to believe that new worms will be released in the future. These worms will be not only more complex, but also more efficient in their propagation, and thus more damaging in their scope.

A new NBAR feature (introduced in Cisco IOS Release 12.3[4]T) enables network administrators to extend the capability of NBAR to classify (and monitor) additional static port applications or to allow NBAR to classify unsupported static port traffic. Specifically, it enables administrators to define the strings that they want to search for in the application payload (for any application, not just HTTP URLs) to identify a given application.

This functionality can be used to identify proprietary applications that otherwise could not be matched. However, it also can be very useful in plugging holes that future worms might open.

For example, consider the example of a fictitious worm called Moonbeam. Moonbeam scans and propagates itself on randomly generated TCP ports within the range of 21000 through 21999. Furthermore, the worm carries the word Moonbeam within the payload, beginning with the ninth ASCII character of the string. Moonbeam's (fictitious) payload is shown in Example 4-17.

### *Example 4-17 Moonbeam Worm Payload*

```
%u[65&%]Moonbeam\x01\x01\x01\x01.*[.][Dd][Ll][Ll]u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%
u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a 403 ...
```

Moonbeam could be identified by a custom NBAR PDLM that examines TCP packets within the range of 21000 through 21999, offsets the scan by 8 ASCII characters, and checks for the string "Moonbeam" (case sensitive), as shown in Example 4-18.

### *Example 4-18 NBAR Policies to Identify Moonbeam*

```
!
ip nbar custom MOONBEAM 8 ascii Moonbeam tcp range 21000 21999  ! "Moonbeam" PDLM
!
class-map match-all MOONBEAM-WORM
 match protocol MOONBEAM        ! Matches the "Moonbeam" custom PDLM
!
```

Verification commands:

- **show policy**
- **show ip nbar port-map**

## Policing Known Worms

These are just a few examples of known worms that can be identified using NBAR. After traffic generated by known worms has been positively identified, it should not be re-marked or limited; rather, it should be dropped immediately. This can be done on ingress on branch LAN edges, as shown in Example 4-19, which combines the policies for Code Red, NIMDA, SQL Slammer, RPC DCOM/W32/MS Blaster, and Sasser. Additionally, the fictitious worm Moonbeam has been included in the policy.

**Note** A recursive classification is required in this policy for SQL Slammer. This is because SQL Slammer requires a **match-all** criteria for its initial classification, but for policy-management purposes, it is desired that this initial classification of SQL Slammer be lumped under a single policy (with a **match-any** criteria) to identify and drop all known worms.

***Example 4-19 NBAR Branch LAN Edge Ingress Policy for Known Worms***

```
!
ip nbar port-map custom-02 udp 1434           ! SQL Slammer custom PDLM
ip nbar port-map custom-03 tcp 5554 9996      ! Sasser custom PDLM
ip nbar port-map netbios tcp 137 139 445      ! MS Blaster TCP 137/139/445
ip nbar port-map netbios udp 135 137 138 139 445 ! MS Blaster UDP 135/137-139/445
ip nbar custom MOONBEAM 8 ascii Moonbeam tcp range 21000 21999  ! "Moonbeam" PDLM
!
class-map match-all SQL-SLAMMER
   match protocol custom-02                  ! Matches the SQL Slammer PDLM
   match packet length min 404 max 404       ! Matches the packet length (376+28)
!
class-map match-any WORMS
   match protocol http url "*.ida*"          ! CodeRed
   match protocol http url "*cmd.exe*"       ! CodeRed
   match protocol http url "*root.exe*"      ! CodeRed
   match protocol http url "*readme.eml*"    ! NIMDA
   match class-map SQL-SLAMMER               ! SQL Slammer class-map
   match protocol exchange                   ! MS Blaster (TCP 135)
   match protocol netbios                    ! MS Blaster NetBIOS PDLM
   match protocol custom-03                  ! Sasser custom PDLM
   match protocol MOONBEAM                   ! "Moonbeam" PDLM
!
policy-map WORM-DROP
 class WORMS
   drop                                       ! Drops all known worms
!
...
!
interface FastEthernet0/0
 no ip address
 speed auto
 duplex auto
!
interface FastEthernet0/0.60
 description DVLAN SUBNET 10.1.60.0
 encapsulation dot1Q 60
 ip address 10.1.60.1 255.255.255.0
 service-policy input WORM-DROP           ! Drops known worms (DVLAN only)
!
```

Verification commands:

- **show policy map**
- **show policy-map interface**
- **show ip nbar port-map**

**Note** For a case study example of Branch Router QoS design, refer to Figure 14-4 and Example 14-20 of the Cisco Press book, *End-to-End QoS Network Design* by Tim Szigeti and Christina Hattingh.

# Summary

Although the QoS design recommendations for branch routers are very similar to and are related to the QoS recommendations for WAN aggregators, this chapter examined four unique considerations of branch routers.

The first consideration was whether to define policies manually or automatically, via the AutoQoS—Enterprise feature. AutoQoS— Enterprise can automatically detect and provision bandwidth for up to 10 classes of traffic and is well suited to smaller WAN/Branch networks managed by administrators with moderate QoS expertise. However for larger WAN/Branch networks, where centralized policies are generally preferred, this feature may not be appropriate due to its limitations.

The second consideration is whether applications provisioned over the WAN are bidirectional or unidirectional. Some unidirectional applications, such as Streaming-Video, provisioned on the WAN aggregator WAN edge do not need to be provisioned correspondingly on the branch router's WAN edge. Bandwidth from unidirectional application classes can be redistributed among other preferential application classes on the branch router's WAN edge.

The third consideration unique to branch routers is that ingress marking might need to be performed on branch-to-campus traffic. This might be because the remote branch access switch does not have the capability to classify and mark traffic, or it might be because some applications require stateful packet inspection (NBAR) to identify them correctly. In either case, ingress marking policies would be required on the branch router's LAN edge, on the data VLAN's subinterface (the voice VLAN traffic markings are trusted). Branch-to-campus traffic can be identified by Layer 3 parameters (such as destination subnet), Layer 4 parameters (such as well-known TCP/UDP ports), or NBAR PDLMs. An example of each type of classification is provided. Additionally, optional DSCP-to-CoS mapping policies (for restoring 802.1p CoS markings that were lost when campus-originated traffic traversed the WAN media) can be set on the branch router's LAN edge.

A fourth unique consideration in branch QoS design is that branch router ingress LAN edges are a strategic place to deploy NBAR policies for worm identification and policing. NBAR policies can be used to identify and drop Code Red, NIMDA, SQL Slammer, RPC DCOM/W32/MS Blaster, Sasser, and other worms. This chapter discussed an extension of the NBAR feature that enables administrators to program the strings that they want NBAR to search packet payloads for; this feature enables NBAR policies to be used to identify new worms that undoubtedly will be released in the future.

# References

## Standards

- RFC 2474 "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
  http://www.ietf.org/rfc/rfc2474
- RFC 2597 "Assured Forwarding PHB Group"
  http://www.ietf.org/rfc/rfc2597
- RFC 3246 "An Expedited Forwarding PHB (Per-Hop Behavior)"
  http://www.ietf.org/rfc/rfc3246

# Books


- Szigeti, Tim and Christina Hattingh. *End-to-End QoS Network Design: Quality of Service in LANs, WANs and VPNs*. Indianapolis: Cisco Press, 2004.


# Cisco IOS Documentation

- Class-based marking (Cisco IOS Release 12.1.5T)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t5/cbpmark2.htm

- Enhanced packet marking (Cisco IOS Release 12.2.13T)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ftenpkmk.htm

- NBAR overview (Cisco IOS Release 12.3)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt1/qcfclass.htm#1003102

- Network-Based Application Recognition (Cisco IOS Release 12.1.5T)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt1/qcfnbar.htm

- Network-Based Application Recognition (Cisco IOS Release 12.2.8T)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t8/dtnbarad.htm

- Network-Based Application Recognition (Cisco IOS Release 12.3[4]T)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t8/dtnbarad.htm

- AutoQoS for the Enterprise (Cisco IOS Release 12.3[11]T)

  http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123t/123t_11/ft_aqose.htm

# Cisco SAFE™ Whitepapers

- SAFE worm mitigation

  http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking_solutions_white_paper09186a00801e120c.shtml

- Using Network-Based Application Recognition and ACLs for blocking the Code Red worm

  http://www.cisco.com/en/US/products/hw/routers/ps359/products_tech_note09186a00800fc176.shtml

- How to protect your network against the NIMDA virus

  http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_tech_note09186a0080110d17.shtml

- SAFE SQL Slammer worm attack mitigation

http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking_solutions_white_paper09186a00801cd7f5.shtml

- SAFE RPC DCOM/W32/Blaster attack mitigation

  http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking_solutions_white_paper09186a00801b2391.shtml

- Combating the Internet worm Sasser

  http://www.cisco.com/application/pdf/en/us/guest/netsol/ns441/c664/cdccont_0900aecd800f613b.pdf