



Context-Based Access Control

This feature module describes the Context-based Access Control (CBAC) feature. It includes information on the benefits of the feature, supported platforms, configuration tasks, and so forth.

This document includes the following sections:

- Feature Overview on page 1
- Supported Platforms on page 10
- Supported Standards, MIBs, and RFCs on page 10
- Configuration Tasks on page 11
- Monitoring and Maintaining CBAC on page 23
- Configuration Examples on page 26
- Command Reference on page 42
- Debug Commands on page 50

Feature Overview

CBAC provides advanced traffic filtering functionality and serves as an integral part of your network's firewall. The information in this document updates the information in the Cisco IOS Release 12.0 *Security Configuration Guide* with the latest feature enhancements:

- Application support for Microsoft NetShow
- IP packet fragmentation attack detection and prevention
- Configurable audit trail and alert messages for CBAC-inspected protocols
- Support for the Cisco IOS Intrusion Detection System (IDS)

For more information regarding firewalls, refer to the chapter “Cisco IOS Firewall Overview” in the Cisco IOS Release 12.0 *Security Configuration Guide*.

For a complete description of the CBAC commands, refer to the “Context-Based Access Control Commands” chapter in the Cisco IOS Release 12.0 *Security Command Reference*.

This section describes:

- What CBAC Does
- What CBAC Does Not Do
- How CBAC Works

- When and Where to Configure CBAC
- The CBAC Process
- Supported Protocols
- Benefits
- Restrictions

What CBAC Does

CBAC works to provide network protection on multiple levels using the following functions:

- Traffic Filtering
- Traffic Inspection
- Alerts and Audit Trails
- Intrusion Detection

Traffic Filtering

CBAC intelligently filters TCP and UDP packets based on application-layer protocol session information. You can configure CBAC to permit specified TCP and UDP traffic through a firewall only when the connection is initiated from within the network you want to protect. CBAC can inspect traffic for sessions that originate from either side of the firewall. CBAC can be used for intranet, extranet, and Internet perimeters of your network. In Cisco IOS Release 12.0(5)T, CBAC provides support for Microsoft's NetShow protocol.

Without CBAC, traffic filtering is limited to access list implementations that examine packets at the network layer, or at most, the transport layer. However, CBAC examines not only network layer and transport layer information but also examines the application-layer protocol information (such as FTP connection information) to learn about the state of the TCP or UDP session. This allows support of protocols that involve multiple channels created as a result of negotiations in the control channel. Most of the multimedia protocols as well as some other protocols (such as FTP, RPC, and SQL*Net) involve multiple channels.

Using CBAC, Java blocking can be configured to filter traffic based on the server address or to completely deny access to Java applets that are not embedded in an archived or compressed file. With Java, you must protect against the risk of users inadvertently downloading destructive applets into your network. To protect against this risk, you could require all users to disable Java in their browser. If this is not an acceptable solution, you can create a CBAC inspection rule to filter Java applets at the firewall, which allows users to download only applets residing within the firewall and trusted applets from outside the firewall. For extensive content filtering of Java, Active-X, or virus scanning, you might want to consider purchasing a dedicated content filtering product.

Traffic Inspection

CBAC inspects traffic that travels through the firewall to discover and manage state information for TCP and UDP sessions. This state information is used to create temporary openings in the firewall's access lists to allow return traffic and additional data connections for permissible sessions (sessions that originated from within the protected internal network).

Inspecting packets at the application layer, and maintaining TCP and UDP session information, provides CBAC with the ability to detect and prevent certain types of network attacks such as SYN-flooding. A SYN-flood attack occurs when a network attacker floods a server with a barrage of requests for connection and does not complete the connection. The resulting volume of half-open connections can overwhelm the server, causing it to deny service to valid requests. Network attacks that deny access to a network device are called denial-of-service (DoS) attacks.

CBAC inspection helps to protect against DoS attacks in other ways. CBAC inspects packet sequence numbers in TCP connections to see if they are within expected ranges—CBAC drops any suspicious packets. You can also configure CBAC to drop half-open connections, which require firewall processing and memory resources to maintain. Additionally, CBAC can detect unusually high rates of new connections and issue alert messages.

CBAC inspection can help protect against certain DoS attacks involving fragmented IP packets. Even though the firewall prevents an attacker from making actual connections to a given host, the attacker can disrupt services provided by that host. This is done by sending many non-initial IP fragments or by sending complete fragmented packets through a router with an ACL that filters the first fragment of a fragmented packet. These fragments can tie up resources on the target host as it tries to reassemble the incomplete packets.

Alerts and Audit Trails

CBAC also generates real-time alerts and audit trails based on events tracked by the firewall. Enhanced audit trail features use SYSLOG to track all network transactions; recording time stamps, source host, destination host, ports used, and the total number of transmitted bytes, for advanced, session-based reporting. Real-time alerts send SYSLOG error messages to central management consoles upon detecting suspicious activity. Using CBAC inspection rules, you can configure alerts and audit trail information on a per-application protocol basis. For example, if you want to generate audit trail information for HTTP traffic, you can specify that in the CBAC rule covering HTTP inspection.

Intrusion Detection

The Cisco IOS Firewall now offers intrusion detection technology for mid-range and high-end router platforms with firewall support. It is ideal for any network perimeter, and especially for locations in which a router is being deployed and additional security between network segments is required. It also can protect intranet and extranet connections where additional security is mandated, and branch-office sites connecting to the corporate office or Internet.

The Cisco IOS Firewall's Intrusion Detection System (Cisco IOS IDS) identifies 59 of the most common attacks using signatures to detect patterns of misuse in network traffic. The intrusion-detection signatures available in the new release of the Cisco IOS Firewall were chosen from a broad cross-section of intrusion-detection signatures. The signatures represent severe breaches of security and the most common network attacks and information-gathering scans.

For more information about Cisco IOS IDS, refer to the document “Configuring Cisco IOS Firewall Intrusion Detection.”

What CBAC Does Not Do

CBAC does not provide intelligent filtering for all protocols; it only works for the protocols that you specify. If you do not specify a certain protocol for CBAC, the existing access lists will determine how that protocol is filtered. No temporary openings will be created for protocols not specified for CBAC inspection.

CBAC does not protect against attacks originating from within the protected network unless that traffic travels through a router that has the Cisco IOS Firewall deployed on it. CBAC only detects and protects against attacks that travel through the firewall. This is a scenario in which you might want to deploy CBAC on an intranet-based router.

CBAC protects against certain types of attacks, but not every type of attack. CBAC should not be considered a perfect, impenetrable defense. Determined, skilled attackers might be able to launch effective attacks. While there is no such thing as a perfect defense, CBAC detects and prevents most of the popular attacks on your network.

How CBAC Works

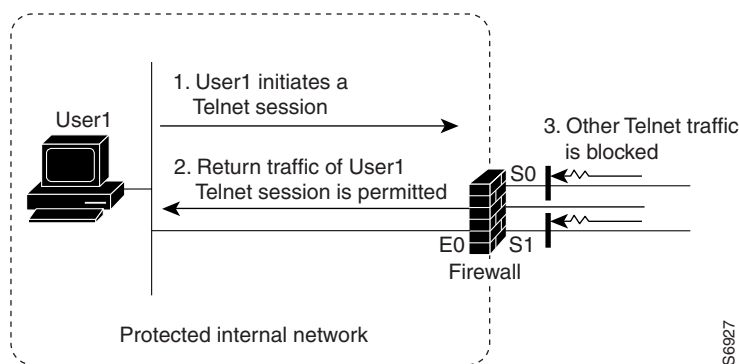
You should understand the material in this section before you configure CBAC. If you do not understand how CBAC works, you might inadvertently introduce security risks by configuring CBAC inappropriately.

How CBAC Works—Overview

CBAC creates temporary openings in access lists at firewall interfaces. These openings are created when specified traffic exits your internal network through the firewall. The openings allow returning traffic (that would normally be blocked) and additional data channels to enter your internal network back through the firewall. The traffic is allowed back through the firewall only if it is part of the same session as the original traffic that triggered CBAC when exiting through the firewall.

In Figure 1, the inbound access lists at S0 and S1 are configured to block Telnet traffic, and there is no outbound access list configured at E0. When the connection request for User1's Telnet session passes through the firewall, CBAC creates a temporary opening in the inbound access list at S0 to permit returning Telnet traffic for User1's Telnet session. (If the same access list is applied to both S0 and S1, the same opening would appear at both interfaces.) If necessary, CBAC would also have created a similar opening in an outbound access list at E0 to permit return traffic.

Figure 1 CBAC Opens Temporary Holes in Firewall Access Lists



S6927

How CBAC Works—Details

This section describes how CBAC inspects packets and maintains state information about sessions to provide intelligent filtering.

Packets Are Inspected

With CBAC, you specify which protocols you want to be inspected, and you specify an interface and interface direction (in or out) where inspection originates. Only specified protocols will be inspected by CBAC. For these protocols, packets flowing through the firewall in any direction are inspected, as long as they flow through the interface where inspection is configured.

Packets entering the firewall are inspected by CBAC only if they first pass the inbound access list at the interface. If a packet is denied by the access list, the packet is simply dropped and not inspected by CBAC.

CBAC inspects and monitors only the control channels of connections; the data channels are not inspected. For example, during FTP sessions both the control and data channels (which are created when a data file is transferred) are monitored for state changes, but only the control channel is inspected (that is, the CBAC software parses the FTP commands and responses).

CBAC inspection recognizes application-specific commands in the control channel, and detects and prevents certain application-level attacks.

CBAC inspection tracks sequence numbers in all TCP packets, and drops those packets with sequence numbers that are not within expected ranges.

CBAC inspection recognizes application-specific commands (such as illegal SMTP commands) in the control channel, and detects and prevents certain application-level attacks.

When CBAC suspects an attack, the DoS feature can take several actions:

- Generate alert messages
- Protect system resources that could impede performance
- Block packets from suspected attackers

CBAC uses timeout and threshold values to manage session state information, helping to determine when to drop sessions that do not become fully established. Setting timeout values for network sessions helps prevent DoS attacks by freeing up system resources, dropping sessions after a specified amount of time. Setting threshold values for network sessions helps prevent DoS attacks by controlling the number of half-open sessions, which limits the amount of system resources applied to half-open sessions. When a session is dropped, CBAC sends a reset message to the devices at both end points (source and destination) of the session. When the system under DoS attack receives a reset command, it releases, or frees up, processes and resources related to that incomplete session.

CBAC provides three thresholds against DoS attacks:

- the total number of half-open TCP or UDP sessions
- the number of half-open sessions based upon time
- the number of half-open TCP-only sessions per host

If a threshold is exceeded, CBAC has two options:

- Send a reset message to the end points of the oldest half-open session, making resources available to service newly arriving SYN packets.
- In the case of half open TCP only sessions, CBAC blocks all SYN packets temporarily for the duration configured by the threshold value. When the router blocks a SYN packet, the TCP three-way handshake is never initiated, which prevents the router from using memory and processing resources needed for valid connections.

DoS detection and prevention requires that you create a CBAC inspection rule and apply that rule on an interface. The inspection rule must include the protocols that you want to monitor against DoS attacks. For example, if you have TCP inspection enabled on the inspection rule, then CBAC can track all TCP connections to watch for DoS attacks. If the inspection rule includes FTP protocol inspection but not TCP inspection, CBAC tracks only FTP connections for DoS attacks.

For detailed information about setting timeout and threshold values in CBAC to detect and prevent DoS attacks, refer in the “Configuring Global Timeouts and Thresholds” section.

A State Table Maintains Session State Information

Whenever a packet is inspected, a state table is updated to include information about the state of the packet’s connection.

Return traffic will only be permitted back through the firewall if the state table contains information indicating that the packet belongs to a permissible session. Inspection controls the traffic that belongs to a valid session and forwards the traffic it does not know. When return traffic is inspected, the state table information is updated as necessary.

UDP “Sessions” Are Approximated

With UDP—a connectionless service—there are no actual sessions, so the software approximates sessions by examining the information in the packet and determining if the packet is similar to other UDP packets (for example, similar source/destination addresses and port numbers) and if the packet was detected soon after another similar UDP packet. “Soon” means within the configurable UDP idle timeout period.

Access List Entries Are Dynamically Created and Deleted

CBAC dynamically creates and deletes access list entries at the firewall interfaces, according to the information maintained in the state tables. These access list entries are applied to the interfaces to examine traffic flowing back into the internal network. These entries create temporary openings in the firewall to permit only traffic that is part of a permissible session.

The temporary access list entries are never saved to NVRAM.

When and Where to Configure CBAC

Configure CBAC at firewalls protecting internal networks. Such firewalls should be Cisco routers with the Cisco Firewall feature set configured as described previously in the section “The Cisco IOS Firewall Feature Set.”

Use CBAC when the firewall will be passing traffic such as:

- Standard TCP and UDP Internet applications
- Multimedia applications
- Oracle support

Use CBAC for these applications if you want the application’s traffic to be permitted through the firewall only when the traffic session is initiated from a particular side of the firewall (usually from the protected internal network).

In many cases, you will configure CBAC in one direction only at a single interface, which causes traffic to be permitted back into the internal network only if the traffic is part of a permissible (valid, existing) session. This is a typical configuration for protecting your internal networks from traffic that originates on the Internet.

You can also configure CBAC in two directions at one or more interfaces. CBAC is configured in two directions when the networks on both sides of the firewall should be protected, such as with extranet or intranet configurations, and to protect against DoS attacks. For example, if the firewall is situated between two partner companies' networks, you might wish to restrict traffic in one direction for certain applications, and restrict traffic in the opposite direction for other applications.

The CBAC Process

This section describes a sample sequence of events that occurs when CBAC is configured at an external interface that connects to an external network such as the Internet.

In this example, a TCP packet exits the internal network through the firewall's external interface. The TCP packet is the first packet of a Telnet session, and TCP is configured for CBAC inspection.

- 1 The packet reaches the firewall's external interface.
- 2 The packet is evaluated against the interface's existing outbound access list, and the packet is permitted. (A denied packet would simply be dropped at this point.)
- 3 The packet is inspected by CBAC to determine and record information about the state of the packet's connection. This information is recorded in a new state table entry created for the new connection.
- 4 (If the packet's application—Telnet—was not configured for CBAC inspection, the packet would simply be forwarded out the interface at this point without being inspected by CBAC. See the section "Define an Inspection Rule" for configuring CBAC inspection information.)
- 5 Based on the obtained state information, CBAC creates a temporary access list entry which is inserted at the beginning of the external interface's inbound extended access list. This temporary access list entry is designed to permit inbound packets that are part of the same connection as the outbound packet just inspected.
- 6 The outbound packet is forwarded out the interface.
- 7 Later, an inbound packet reaches the interface. This packet is part of the same Telnet connection previously established with the outbound packet. The inbound packet is evaluated against the inbound access list, and it is permitted because of the temporary access list entry previously created.
- 8 The permitted inbound packet is inspected by CBAC, and the connection's state table entry is updated as necessary. Based on the updated state information, the inbound extended access list temporary entries might be modified in order to permit only packets that are valid for the current state of the connection.
- 9 Any additional inbound or outbound packets that belong to the connection are inspected to update the state table entry and to modify the temporary inbound access list entries as required, and they are forwarded through the interface.
- 10 When the connection terminates or times out, the connection's state table entry is deleted, and the connection's temporary inbound access list entries are deleted.

In the sample process just described, the firewall access lists are configured as follows:

- An outbound IP access list (standard or extended) is applied to the external interface. This access list permits all packets that you want to allow to exit the network, including packets you want to be inspected by CBAC. In this case, Telnet packets are permitted.
- An inbound extended IP access list is applied to the external interface. This access list denies any traffic to be inspected by CBAC—including Telnet packets. When CBAC is triggered with an outbound packet, CBAC creates a temporary opening in the inbound access list to permit only traffic that is part of a valid, existing session.

If the inbound access list had been configured to permit *all* traffic, CBAC would be creating pointless openings in the firewall for packets that would be permitted anyway.

Supported Protocols

You can configure CBAC to inspect the following types of sessions:

- All TCP sessions, regardless of the application-layer protocol (sometimes called “single-channel” or “generic” TCP inspection)
- All UDP sessions, regardless of the application-layer protocol (sometimes called “single-channel” or “generic” UDP inspection)

You can also configure CBAC to specifically inspect certain application-layer protocols. The following application-layer protocols can all be configured for CBAC:

- CU-SeeMe (only the White Pine version)
- FTP
- H.323 (such as NetMeeting, ProShare)
- HTTP (Java blocking)
- Java
- Microsoft NetShow
- UNIX R-commands (such as rlogin, rexec, and rsh)
- RealAudio
- RPC (Sun RPC, not DCE RPC)
- Microsoft RPC
- SMTP
- SQL*Net
- StreamWorks
- TFTP
- VDOLive

When a protocol is configured for CBAC, that protocol traffic is inspected, state information is maintained, and in general, packets are allowed back through the firewall only if they belong to a permissible session.

Benefits

CBAC provides the following features:

- Stateful packet filtering—Provides sophisticated security and policy enforcement for connections within an organization (intranet) and between an organization and its partner networks, as well as between the organization and the Internet.
- Denial-of-service detection and prevention— CBAC defends and protects router resources against common attacks, checking packet headers and dropping suspicious packets.
- Real-time alerts and audit trail information—Configurable on a per-application basis to track connection information for traffic through the firewall, providing detailed usage information and reporting on suspicious activity.
- Seamless interoperability—Integrates the firewall solution with other Cisco IOS software features; optimizing WAN utilization; providing robust, scalable routing; and interoperating with existing Cisco IOS-based networks (such as the Internet).
- VPN Support—Using Cisco IOS Firewall with other Cisco IOS encryption and Quality of Service (QoS) features enables secure, low-cost transmission over public networks, reduces implementation and management total cost of ownership for remote branch offices and extranets, and insures mission-critical application traffic receives high priority delivery.
- Scalable deployment—Available for a wide variety of router platforms, the Cisco IOS Firewall scales to meet any network's bandwidth and performance requirements.
- Java blocking—Protects against unidentified, malicious Java applets.

Restrictions

- CBAC is available only for IP protocol traffic. Only TCP and UDP packets are inspected. (Other IP traffic, such as ICMP, cannot be inspected with CBAC and should be filtered with basic access lists instead.)
- If you reconfigure your access lists when you configure CBAC, be aware that if your access lists block TFTP traffic into an interface, you will not be able to netboot over that interface. (This is not a CBAC-specific limitation, but is part of existing access list functionality.)
- Packets with the firewall as the source or destination address are not inspected by CBAC or evaluated by access lists.
- CBAC ignores ICMP Unreachable messages.
- With FTP, CBAC does not allow third-party connections (three-way FTP transfer).
- When CBAC inspects FTP traffic, it only allows data channels with the destination port in the range of 1024 to 65535.
- CBAC will not open a data channel if the FTP client-server authentication fails.
- Cisco Encryption Technology (CET) and CBAC compatibility.
 - If encrypted traffic is exchanged between two routers, and the firewall is in between the two routers, CBAC might not work as anticipated. This is because the packets' payloads are encrypted, so CBAC cannot accurately inspect the payloads.
 - Also, if both encryption and CBAC are configured at the same firewall, CBAC will not work for certain protocols. In this case, CBAC will work with single-channel TCP and UDP, except for Java and SMTP. But CBAC will not work with multichannel protocols, except for

StreamWorks and CU-SeeMe. So if you configure encryption at the firewall, you should configure CBAC for only the following protocols: Generic TCP, Generic UDP, CU-SeeMe, StreamWorks.

- IPsec and CBAC compatibility
 - When CBAC and IPsec are enabled on the same router, and the firewall router is an end point for IPsec for the particular flow, then IPsec is compatible with CBAC (that is, CBAC can do its normal inspection processing on the flow).
 - If the router is not an IPsec end point, but the packet is an IPsec packet, then CBAC will not inspect the packets because the protocol number in the IP header of the IPsec packet is not TCP or UDP. CBAC only inspects TCP and UDP packets.

Supported Platforms

The Cisco IOS Firewall feature set is supported on the following platforms:

- Cisco 800 series
- Cisco uBR900 series
- Cisco 1600 series
- Cisco 1700 series
- Cisco 2500 series
- Cisco 2600 series
- Cisco 3600 series
- Cisco 7100 series
- Cisco 7200 series

Supported Standards, MIBs, and RFCs

MIBs

No new or modified MIBs are supported by this feature.

For descriptions of supported MIBs and how to use MIBs, see the Cisco MIB web site on CCO at <http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml>.

RFCs

No new or modified RFCs are supported by this feature.

Standards

No new or modified standards are supported by this feature.

Configuration Tasks

See the following sections for configuration tasks for CBAC. Each task in the list indicates if the task is optional or required.

- Picking an Interface: Internal or External (Required)
- Configuring IP Access Lists at the Interface (Required)
- Configuring Global Timeouts and Thresholds (Required)
- Defining an Inspection Rule (Required)
- Applying the Inspection Rule to an Interface (Required)
- Configuring Logging and Audit Trail (Required)
- Other Guidelines for Configuring a Firewall (Required)
- Verifying CBAC (Optional)

Note If you try to configure Context-based Access Control (CBAC) but do not have a good understanding of how CBAC works, you might inadvertently introduce security risks to the firewall and to the protected network. Be sure you understand what CBAC does before you configure CBAC.

For CBAC configuration examples, refer to the “Configuration Examples” section.

Picking an Interface: Internal or External

You must decide whether to configure CBAC on an internal or external interface of your firewall.

“Internal” refers to the side where sessions must originate for their traffic to be permitted through the firewall. “External” refers to the side where sessions cannot originate (sessions originating from the external side will be blocked).

If you will be configuring CBAC in two directions, you should configure CBAC in one direction first, using the appropriate “internal” and “external” interface designations. When you configure CBAC in the other direction, the interface designations will be swapped. (CBAC can be configured in two directions at one or more interfaces. Configure CBAC in two directions when the networks on both sides of the firewall require protection, such as with extranet or intranet configurations, and for protection against DoS attacks.)

The firewall is most commonly used with one of two basic network topologies. Determining which of these topologies is most like your own can help you decide whether to configure CBAC on an internal interface or on an external interface.

Figure 2 shows the first network topology. In this simple topology, CBAC is configured for the *external* interface Serial 1. This prevents specified protocol traffic from entering the firewall and the internal network, unless the traffic is part of a session initiated from within the internal network.

Figure 2 Simple Topology—CBAC Configured at the External Interface

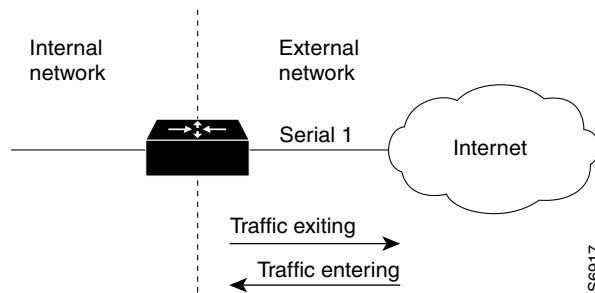
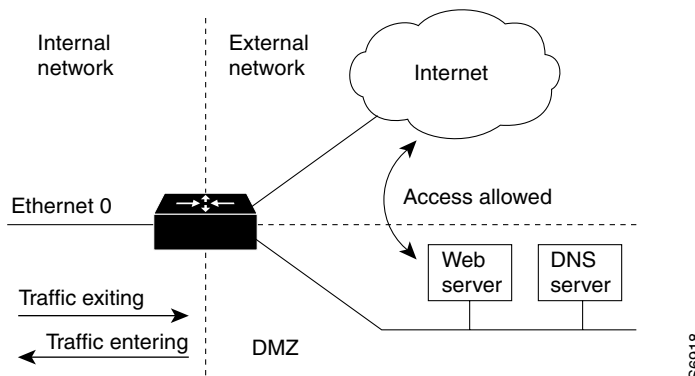


Figure 3 shows the second network topology. In this topology, CBAC is configured for the *internal* interface Ethernet 0. This allows external traffic to access the services in the Demilitarized Zone (DMZ), such as DNS services, but prevents specified protocol traffic from entering your internal network—unless the traffic is part of a session initiated from within the internal network.

Figure 3 DMZ Topology—CBAC Configured at the Internal Interface



Using these two sample topologies, decide whether to configure CBAC on an internal or external interface.

Configuring IP Access Lists at the Interface

For CBAC to work properly, you need to make sure that you have IP access lists configured appropriately at the interface.

Follow these three general rules when evaluating your IP access lists at the firewall:

- Start with a basic configuration.

If you try to configure access lists without a good understanding of how access lists work, you might inadvertently introduce security risks to the firewall and to the protected network. You should be sure you understand what access lists do before you configure your firewall. For more information about access control lists, refer to the “Access Control Lists: Overview and Guidelines” chapter of the Cisco IOS Release 12.0 *Security Configuration Guide*.

A basic initial configuration allows all network traffic to flow from the protected networks to the unprotected networks, while blocking network traffic from any unprotected networks.

- Permit CBAC traffic to leave the network through the firewall.

All access lists that evaluate traffic leaving the protected network should permit traffic that will be inspected by CBAC. For example, if Telnet will be inspected by CBAC, then Telnet traffic should be permitted on all access lists that apply to traffic leaving the network.

- Use extended access lists to deny CBAC return traffic entering the network through the firewall.

For temporary openings to be created in an access list, the access list must be an extended access list. So wherever you have access lists that will be applied to returning traffic, you must use extended access lists. The access lists should deny CBAC return traffic because CBAC will open up temporary holes in the access lists. (You want traffic to be normally blocked when it enters your network.)

Note If your firewall only has two connections, one to the internal network and one to the external network, using all inbound access lists works well because packets are stopped before they get a chance to affect the router itself.

Basic Configuration

The first time you configure the Cisco IOS Firewall, it is helpful to start with a basic access list configuration that makes the operation of the firewall easy to understand without compromising security. The basic configuration allows all network traffic from the protected networks access to the unprotected networks, while blocking all network traffic (with some exceptions) from the unprotected networks to the protected networks.

Any firewall configuration depends on your site security policy. If the basic configuration does not meet your initial site security requirements, configure the firewall to meet your policy. If you are unfamiliar with that policy or need help with the configuration, contact your network administration group for assistance. For additional guidelines on configuring a firewall, refer to “Other Guidelines for Configuring a Firewall” on page 21.

Use these guidelines for configuring the initial firewall access lists:

- Do not configure an access list for traffic from the protected networks to the unprotected networks, meaning that all traffic from the protected networks can flow through the interface.

This helps to simplify firewall management by reducing the number of access lists applied at the interfaces. Of course this assumes a high level of trust for the users on the protected networks, and it assumes there are no malicious users on the protected networks who might launch attacks from the “inside.” You can fine tune network access for users on the protected networks as you gain experience with access list configuration and the operation of the firewall.

- Configure an access list that includes entries permitting certain ICMP traffic from unprotected networks.

While an access list that denies all IP traffic not part of a connection inspected by CBAC seems most secure, it is not practical for normal operation of the router. The router expects to see ICMP traffic from other routers in the network. Additionally, ICMP traffic is not inspected by CBAC, meaning specific entries are needed in the access list to permit return traffic for ICMP commands. For example, a user on a protected network uses the **ping** command to get the status of a host on an unprotected network; without entries in the access list that permit **echo reply** messages, the user on the protected network gets no response to the **ping** command.

Include access list entries to permit the following ICMP messages:

Message	Description
echo reply	Outgoing ping commands require echo-reply messages to come back.
time-exceeded	Outgoing traceroute commands require time-exceeded messages to come back.
packet-too-big	Path MTU discovery requires “too-big” messages to come back.
traceroute	Allow an incoming traceroute.
unreachable	Permit all “unreachable” messages to come back. If a router cannot forward or deliver a datagram, it sends an ICMP unreachable message back to the source and drops the datagram.

- Add an access list entry denying any network traffic from a source address matching an address on the protected network.

This is known as anti-spoofing protection because it prevents traffic from an unprotected network from assuming the identity of a device on the protected network.

- Add an entry denying broadcast messages with a source address of 255.255.255.255.

This entry helps to prevent broadcast attacks.

- By default, the last entry in an extended access list is an implicit denial of all IP traffic not specifically allowed by other entries in the access list.

Although this is the default setting, this final deny statement is not shown by default in an access list. Optionally, you can add an entry to the access list denying IP traffic with any source or destination address with no undesired effects.

For complete information about how to configure IP access lists, refer to the “Configuring IP Services” chapter of the Cisco IOS Release 12.0 *Network Protocols Configuration Guide, Part 1*.

For tips on applying access lists at an external or internal interface, review the sections “External Interface” and “Internal Interface” in this document.

External Interface

Here are some tips for your access lists when you will be configuring CBAC on an external interface:

- If you have an outbound IP access list at the external interface, the access list can be a standard or extended access list. This outbound access list should permit traffic that you want to be inspected by CBAC. If traffic is not permitted, it will not be inspected by CBAC, but will be simply dropped.
- The inbound IP access list at the external interface must be an extended access list. This inbound access list should deny traffic that you want to be inspected by CBAC. (CBAC will create temporary openings in this inbound access list as appropriate to permit only return traffic that is part of a valid, existing session.)
- For complete information about how to configure IP access lists, refer to the “Configuring IP Services” chapter of the Cisco IOS Release 12.0 *Network Protocols Configuration Guide, Part 1*.

Internal Interface

Here are some tips for your access lists when you will be configuring CBAC on an internal interface:

- If you have an inbound IP access list at the internal interface or an outbound IP access list at external interface(s), these access lists can be either a standard or extended access list. These access lists should permit traffic that you want to be inspected by CBAC. If traffic is not permitted, it will not be inspected by CBAC, but will be simply dropped.
- The outbound IP access list at the internal interface and the inbound IP access list at the external interface must be extended access lists. These outbound access lists should deny traffic that you want to be inspected by CBAC. (CBAC will create temporary openings in these outbound access lists as appropriate to permit only return traffic that is part of a valid, existing session.) You do not necessarily need to configure an extended access list at both the outbound internal interface and the inbound external interface, but at least one is necessary to restrict traffic flowing through the firewall into the internal protected network.
- For complete information about how to configure IP access lists, refer to the “Configuring IP Services” chapter of the Cisco IOS Release 12.0 *Network Protocols Configuration Guide, Part 1*.

Configuring Global Timeouts and Thresholds

CBAC uses timeouts and thresholds to determine how long to manage state information for a session, and to determine when to drop sessions that do not become fully established. These timeouts and thresholds apply globally to all sessions.

You can use the default timeout and threshold values, or you can change to values more suitable to your security requirements. You should make any changes to the timeout and threshold values before you continue configuring CBAC.

Note If you want to enable the more aggressive TCP host-specific denial-of-service prevention that includes the blocking of connection initiation to a host, you must set the **block-time** specified in the **ip inspect tcp max-incomplete host** command (see the last row in Table 1).

All the available CBAC timeouts and thresholds are listed in Table 1, along with the corresponding command and default value.

To change a global timeout or threshold listed in the “Timeout or Threshold Value to Change” column, use the global configuration command in the “Command” column:

Table 1 Timeout and Threshold Values

Timeout or Threshold Value to Change	Command	Default
The length of time the software waits for a TCP session to reach the established state before dropping the session.	ip inspect tcp synwait-time <i>seconds</i>	30 seconds
The length of time a TCP session will still be managed after the firewall detects a FIN-exchange.	ip inspect tcp finwait-time <i>seconds</i>	5 seconds
The length of time a TCP session will still be managed after no activity (the TCP idle timeout). ¹	ip inspect tcp idle-time <i>seconds</i>	3600 seconds (1 hour)

Table 1 Timeout and Threshold Values (continued)

Timeout or Threshold Value to Change	Command	Default
The length of time a UDP session will still be managed after no activity (the UDP idle timeout). ¹	ip inspect udp idle-time <i>seconds</i>	30 seconds
The length of time a DNS name lookup session will still be managed after no activity.	ip inspect dns-timeout <i>seconds</i>	5 seconds
The number of existing half-open sessions that will cause the software to start deleting half-open sessions. ²	ip inspect max-incomplete high <i>number</i>	500 existing half-open sessions
The number of existing half-open sessions that will cause the software to stop deleting half-open sessions. ²	ip inspect max-incomplete low <i>number</i>	400 existing half-open sessions
The rate of new sessions that will cause the software to start deleting half-open sessions. ²	ip inspect one-minute high <i>number</i>	500 half-open sessions per minute
The rate of new sessions that will cause the software to stop deleting half-open sessions. ²	ip inspect one-minute low <i>number</i>	400 half-open sessions per minute
The number of existing half-open TCP sessions with the same destination host address that will cause the software to start dropping half-open sessions to the same destination host address. ³	ip inspect tcp max-incomplete host <i>number</i> block-time <i>minutes</i>	50 existing half-open TCP sessions; 0 minutes

1 The global TCP and UDP idle timeouts can be overridden for specified application-layer protocols' sessions as described in the **ip inspect name (global configuration)** command description, found in the "Context-Based Access Control Commands" chapter of the Cisco IOS Release 12.0 *Security Command Reference*.

2 See the following section, "Half-Open Sessions," for more information.

3 Whenever the **max-incomplete host** threshold is exceeded, the software will drop half-open sessions differently depending on whether the **block-time** timeout is zero or a positive non-zero number. If the **block-time** timeout is zero, the software will delete the oldest existing half-open session for the host for every new connection request to the host and will let the SYN packet through. If the **block-time** timeout is greater than zero, the software will delete all existing half-open sessions for the host, and then block all new connection requests to the host. The software will continue to block all new connection requests until the **block-time** expires.

To reset any threshold or timeout to the default value, use the **no** form of the command in Table 1.

Half-Open Sessions

An unusually high number of half-open sessions (either absolute or measured as the arrival rate) could indicate that a denial-of-service attack is occurring. For TCP, "half-open" means that the session has not reached the established state—the TCP three-way handshake has not yet been completed. For UDP, "half-open" means that the firewall has detected no return traffic.

CBAC measures both the total number of existing half-open sessions and the rate of session establishment attempts. Both TCP and UDP half-open sessions are counted in the total number and rate measurements. Measurements are made once a minute.

When the number of existing half-open sessions rises above a threshold (the **max-incomplete high** number), the software will delete half-open sessions as required to accommodate new connection requests. The software will continue to delete half-open requests as necessary, until the number of existing half-open sessions drops below another threshold (the **max-incomplete low** number).

When the rate of new connection attempts rises above a threshold (the **one-minute high** number), the software will delete half-open sessions as required to accommodate new connection attempts. The software will continue to delete half-open sessions as necessary, until the rate of new connection attempts drops below another threshold (the **one-minute low** number). The rate thresholds are measured as the number of new session connection attempts detected in the last one-minute sample period. The firewall router reviews the “one-minute” rate on an ongoing basis, meaning that the router reviews the rate more frequently than one minute and does not keep deleting half-open sessions for one-minute after a DoS attack has stopped—it will be less time.

Defining an Inspection Rule

After you configure global timeouts and thresholds, you must define an inspection rule. This rule specifies what IP traffic (which application-layer protocols) will be inspected by CBAC at an interface.

Normally, you define only one inspection rule. The only exception might occur if you want to enable CBAC in two directions as described earlier in the section “When and Where to Configure CBAC.” For CBAC configured in both directions at a single firewall interface, you should configure two rules, one for each direction.

An inspection rule should specify each desired application-layer protocol as well as generic TCP or generic UDP if desired. The inspection rule consists of a series of statements each listing a protocol and specifying the same inspection rule name.

Inspection rules include options for controlling alert and audit trail messages and for checking IP packet fragmentation.

To define an inspection rule, follow the instructions in the following sections:

- Configuring Application-layer Protocol Inspection
- Configuring Generic TCP and UDP Inspection

Configuring Application-layer Protocol Inspection

This section provides instructions for configuring CBAC with the following inspection information:

- Configuring Application-layer Protocols
- Configuring Java Inspection
- Configuring IP Packet Fragmentation Inspection

Note For CBAC inspection to work with NetMeeting 2.0 traffic (an H.323 application-layer protocol), you must also configure inspection for TCP, as described later in the section “Configuring Generic TCP and UDP Inspection.” This requirement exists because NetMeeting 2.0 uses an additional TCP channel not defined in the H.323 specification.

Configuring Application-layer Protocols

To configure CBAC inspection for an application-layer protocol, use one or both of the following global configuration commands:

Command	Purpose
Router(config)# ip inspect name <i>inspection-name</i> <i>protocol</i> [alert { on off }] [audit-trail { on off }] [timeout <i>seconds</i>]	Configure CBAC inspection for an application-layer protocol (except for RPC and Java). Use one of the protocol keywords defined in Table 2. Repeat this command for each desired protocol. Use the same <i>inspection-name</i> to create a single inspection rule.
Router(config)# ip inspect name <i>inspection-name</i> rpc program-number <i>number</i> [wait-time <i>minutes</i>] [alert { on off }] [audit-trail { on off }] [timeout <i>seconds</i>]	Enable CBAC inspection for the RPC application-layer protocol. You can specify multiple RPC program numbers by repeating this command for each program number. Use the same <i>inspection-name</i> to create a single inspection rule.

Refer to the description of the **ip inspect name (global configuration)** command in the “Command Reference” section on page 42 in this document for complete information about how the command works with various application-layer protocols.

Table 2 identifies application protocol keywords.

Table 2 Application Protocol Keywords

Application Protocol	<i>protocol</i> Keyword
CU-SeeMe	cuseeme
FTP	ftp
H.323	h323
Microsoft NetShow	netshow
UNIX R commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio
SMTP	smtp
RPC	rpc
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Configuring Java Inspection

Java applet filtering distinguishes between trusted and untrusted applets by relying on a list of external sites that you designate as “friendly.” If an applet is from a friendly site, the firewall allows the applet through. If the applet is not from a friendly site, the applet will be blocked. (Alternately, you could permit applets from all external sites except for those you specifically designate as hostile.)

To block all Java applets except for applets from friendly locations, use the following global configuration commands:

Step	Command	Purpose
1	<pre>Router(config)# ip access-list standard name permit ... deny ... (Use permit and deny statements as appropriate.) or Router(config)# access-list access-list-number {deny permit} source [source-wildcard]</pre>	<p>Create a standard access list that permits traffic only from friendly sites, and denies traffic from hostile sites.</p> <p>If you want all internal users to be able to download friendly applets, use the any keyword for the destination as appropriate—but be careful to not misuse the any keyword to inadvertently allow all applets through.</p>
2	<pre>Router(config)# ip inspect name inspection-name http [java-list access-list] [alert {on off}] [audit-trail {on off}] [timeout seconds]</pre>	<p>Block all Java applets except for applets from the friendly sites defined previously in the access list. Java blocking only works with standard access lists.</p> <p>Use the same <i>inspection-name</i> as when you specified other protocols, to create a single inspection rule.</p>



Caution CBAC does not detect or block encapsulated Java applets. Therefore, Java applets that are wrapped or encapsulated, such as applets in .zip or .jar format, are *not* blocked at the firewall. CBAC also does not detect or block applets loaded from FTP, gopher, HTTP on a nonstandard port, and so forth.

Configuring IP Packet Fragmentation Inspection

CBAC inspection rules can help protect hosts against certain DoS attacks involving fragmented IP packets.

Using fragmentation inspection, the firewall maintains an *interfragment state* (structure) for IP traffic. Non-initial fragments are discarded unless the corresponding initial fragment was permitted to pass through the firewall. Non-initial fragments received before the corresponding initial fragments are discarded.

Note Fragmentation inspection can have undesirable effects in certain cases, because it can result in the firewall discarding any packet whose fragments arrive out of order. There are many circumstances that can cause out-of-order delivery of legitimate fragments. Applying fragmentation inspection in situations where legitimate fragments, which are likely to arrive out of order, might have a severe performance impact.

Because routers running Cisco IOS software are used in a large variety of networks, and because the CBAC feature is often used to isolate parts of internal networks from one another, the fragmentation inspection feature is disabled by default. Fragmentation detection must be explicitly enabled for an inspection rule using the **ip inspect name** command. Unfragmented traffic is never discarded because it lacks a fragment state. Even when the system is under heavy attack with fragmented packets, legitimate fragmented traffic, if any, gets some fraction of the firewall's fragment state resources, and legitimate, unfragmented traffic can flow through the firewall unimpeded.

Applying the Inspection Rule to an Interface

To configure CBAC inspection rules for IP fragmentation checking, use the following form of the **ip inspect name** global configuration command:

Command	Purpose
Router(config)# ip inspect name <i>inspection-name</i> fragment [max number timeout number]	Configure IP fragmentation checking in CBAC inspection rules.

Repeat this command for each named inspection rule in which you want to inspect IP fragments.

Configuring Generic TCP and UDP Inspection

You can configure TCP and UDP inspection to permit TCP and UDP packets to enter the internal network through the firewall, even if the application-layer protocol is not configured to be inspected. However, TCP and UDP inspection do not recognize application-specific commands, and therefore might not permit all return packets for an application, particularly if the return packets have a different port number than the previous exiting packet.

Any application-layer protocol that is inspected will take precedence over the TCP or UDP packet inspection. For example, if inspection is configured for FTP, all control channel information will be recorded in the state table, and all FTP traffic will be permitted back through the firewall if the control channel information is valid for the state of the FTP session. The fact that TCP inspection is configured is irrelevant to the FTP state information.

With TCP and UDP inspection, packets entering the network must exactly match the corresponding packet that previously exited the network. The entering packets must have the same source/destination addresses and source/destination port numbers as the exiting packet (but reversed); otherwise, the entering packets will be blocked at the interface. Also, all TCP packets with a sequence number outside of the window are dropped.

With UDP inspection configured, replies will only be permitted back in through the firewall if they are received within a configurable time after the last request was sent out. (This time is configured with the **ip inspect udp idle-time** command.)

To configure CBAC inspection for TCP or UDP packets, use one or both of the following global configuration commands:

Command	Purpose
Router(config)# ip inspect name <i>inspection-name</i> tcp [timeout seconds]	Enable CBAC inspection for TCP packets. Use the same <i>inspection-name</i> as when you specified other protocols, to create a single inspection rule.
Router(config)# ip inspect name <i>inspection-name</i> udp [timeout seconds]	Enable CBAC inspection for UDP packets. Use the same <i>inspection-name</i> as when you specified other protocols, to create a single inspection rule.

Applying the Inspection Rule to an Interface

After you define an inspection rule, you apply that rule to an interface.

Normally, you apply only one inspection rule to one interface. The only exception might occur if you want to enable CBAC in two directions as described earlier in the section “When and Where to Configure CBAC.” For CBAC configured in both directions at a single firewall interface, you should apply two rules, one for each direction.

To apply an inspection rule to an interface, use the following interface configuration command:

Command	Purpose
Router(config-if)# ip inspect <i>inspection-name</i> { in out }	Apply an inspection rule to an interface.

Configuring Logging and Audit Trail

Turn on logging and audit trail to provide a record of network access through the firewall, including illegitimate access attempts, and inbound and outbound services. To configure logging and audit trail functions, enter the following commands in global configuration mode:

Command	Purpose
Router(config)# service timestamps log datetime	Add the date and time to syslog and audit trail messages.
Router(config)# logging <i>host</i>	Specify the host name or IP address of the host where you want to send syslog messages.
Router(config)# logging facility <i>facility-type</i>	Configure the syslog facility in which error messages are sent.
Router(config)# logging trap <i>level</i>	(Optional) Use this command to limit messages logged to the syslog servers based on severity. The default is level 7 (informational).
Router(config)# ip inspect audit-trail	Turn on CBAC audit trail messages.

For information on how to interpret the syslog and audit trail messages, refer to “Interpreting Syslog and Console Messages Generated by CBAC” section on page 23.

To configure audit trail functions on a per-application basis, refer to “Defining an Inspection Rule” on page 17 for more information.

For complete information about how to configure logging, refer to the “Troubleshooting the Router” chapter of the Cisco IOS Release 12.0 *Configuration Fundamentals Configuration Guide*.

Other Guidelines for Configuring a Firewall

As with all networking devices, you should always protect access into the firewall by configuring passwords as described in the “Configuring Passwords and Privileges” chapter in the Cisco IOS Release 12.0 *Security Configuration Guide*. You should also consider configuring user authentication, authorization, and accounting as described in the “Authentication, Authorization, and Accounting (AAA)” part of the Cisco IOS Release 12.0 *Security Configuration Guide*.

You should also consider the following recommendations:

- When setting passwords for privileged access to the firewall, use the **enable secret** command rather than the **enable password** command, which does not have as strong an encryption algorithm.
- Put a password on the console port. In authentication, authorization, and accounting (AAA) environments, use the same authentication for the console as for elsewhere. In a non-AAA environment, at a minimum configure the **login** and **password** *password* commands.
- Think about access control *before* you connect a console port to the network in any way, including attaching a modem to the port. Be aware that a *break* on the console port might give total control of the firewall, even with access control configured.
- Apply access lists and password protection to all virtual terminal ports. Use access lists to limit who can Telnet into your router.

- Do not enable any local service (such as SNMP or NTP) that you do not use. Cisco Discovery Protocol (CDP) and Network Time Protocol (NTP) are on by default, and you should turn these off if you do not need them.

To turn off CDP, enter the **no cdp run** global configuration command. To turn off NTP, enter the **ntp disable** interface configuration command on each interface not using NTP.

If you must run NTP, configure NTP only on required interfaces, and configure NTP to listen only to certain peers.

Any enabled service could present a potential security risk. A determined, hostile party might be able to find creative ways to misuse the enabled services to access the firewall or the network.

For local services that are enabled, protect against misuse. Protect by configuring the services to communicate only with specific peers, and protect by configuring access lists to deny packets for the services at specific interfaces.

- Protect against spoofing: protect the networks on both sides of the firewall from being spoofed from the other side. You could protect against spoofing by configuring input access lists at all interfaces to pass only traffic from expected source addresses, and to deny all other traffic.

You should also disable source routing. For IP, enter the **no ip source-route** global configuration command. Disabling source routing at *all* routers can also help prevent spoofing.

You should also disable minor services. For IP, enter the **no service tcp-small-servers** and **no service udp-small-servers** global configuration commands. In Cisco IOS Release 12.0 and later, these services are disabled by default.

- Prevent the firewall from being used as a relay by configuring access lists on any asynchronous Telnet ports.
- Normally, you should disable directed broadcasts for all applicable protocols on your firewall and on all your other routers. For IP, use the **no ip directed-broadcast** command. Rarely, some IP networks do require directed broadcasts; if this is the case, do not disable directed broadcasts.
Directed broadcasts can be misused to multiply the power of denial-of-service attacks, because every denial-of-service packet sent is broadcast to every host on a subnet. Furthermore, some hosts have other intrinsic security risks present when handling broadcasts.
- Configure the **no proxy-arp** command to prevent internal addresses from being revealed. (This is important to do if you do not already have NAT configured to prevent internal addresses from being revealed).
- Keep the firewall in a secured (locked) room.

Verifying CBAC

You can verify CBAC information by using one or more of the following EXEC commands:

Command	Purpose
Router# show ip inspect name <i>inspection-name</i>	Show a particular configured inspection rule.
Router# show ip inspect config	Show the complete CBAC inspection configuration.
Router# show ip inspect interfaces	Show interface configuration with regards to applied inspection rules and access lists.
Router# show ip inspect session [detail]	Show existing sessions that are currently being tracked and inspected by CBAC.
Router# show ip inspect all	Show all CBAC configuration and all existing sessions that are currently being tracked and inspected by CBAC.

Monitoring and Maintaining CBAC

You can watch for network attacks and investigate network problems using system messages and debug commands.

Interpreting Syslog and Console Messages Generated by CBAC

CBAC provides syslog messages, console alert messages, and audit trail messages. These messages are useful because they can alert you to network attacks, and because they provide an audit trail that provides details about sessions inspected by CBAC. Audit trail and alert information is configurable on a per-application basis using the CBAC inspection rules.

The following types of messages can be generated by CBAC:

- Denial-of-service Messages
- SMTP Messages
- Java Blocking Messages
- FTP Messages
- Audit Trail Messages

For explanations and recommended actions related to the error messages mentioned in this section, refer to the Cisco IOS *Software System Error Messages*.

Denial-of-service Messages

CBAC detects and blocks denial-of-service attacks and notifies you when denial-of-service attacks occur. Error messages such as the following may indicate that denial-of-service attacks have occurred:

```
%FW-4-ALERT_ON: getting aggressive, count (550/500) current 1-min rate: 250
%FW-4-ALERT_OFF: calming down, count (0/400) current 1-min rate: 0
```

When %FW-4-ALERT_ON and %FW-4-ALERT_OFF error messages appear together, each “aggressive/calming” pair of messages indicates a separate attack. The previous example shows one separate attack.

Error messages such as the following may indicate that a denial-of-service attack has occurred on a specific TCP host:

```
%FW-4-HOST_TCP_ALERT_ON: Max tcp half-open connections (50) exceeded for host
172.21.127.242.
%FW-4-BLOCK_HOST: Blocking new TCP connections to host 172.21.127.242 for 2 minutes
(half-open count 50 exceeded)
%FW-4-UNBLOCK_HOST: New TCP connections to host 172.21.127.242 no longer blocked
```

SMTP Messages

CBAC detects and blocks SMTP attacks (illegal SMTP commands) and notifies you when SMTP attacks occur. Error messages such as the following may indicate that an SMTP attack has occurred:

```
%FW-4-SMTP_INVALID_COMMAND: Invalid SMTP command from initiator (192.168.12.3:52419)
```

Java Blocking Messages

CBAC detects and selectively blocks Java applets and notifies you when a Java applet has been blocked. Error messages such as the following may indicate that a Java applet has been blocked:

```
%FW-4-HTTP_JAVA_BLOCK: JAVA applet is blocked from (172.21.127.218:80) to
(172.16.57.30:44673) .
```

FTP Messages

CBAC detects and prevents certain FTP attacks and notifies you when this occurs. Error messages such as the following may appear when CBAC detects these FTP attacks:

```
%FW-3-FTP_PRIV_PORT: Privileged port 1000 used in PORT command -- FTP client 10.0.0.1
FTP server 10.1.0.1
%FW-3-FTP_SESSION_NOT_AUTHENTICATED: Command issued before the session is authenticated
-- FTP client 10.0.0.1
%FW-3-FTP_NON_MATCHING_IP_ADDR: Non-matching address 172.19.148.154 used in PORT
command -- FTP client 172.19.54.143 FTP server 172.16.127.242
```

Audit Trail Messages

CBAC provides audit trail messages to record details about inspected sessions. Audit trail information is configurable on a per-application basis using the CBAC inspection rules. To determine which protocol was inspected, use the responder's port number. The port number follows the responder's address. The following are sample audit trail messages:

```
%FW-6-SESS_AUDIT_TRAIL: tcp session initiator (192.168.1.13:33192) sent 22 bytes --
responder (192.168.129.11:25) sent 208 bytes
%FW-6-SESS_AUDIT_TRAIL: http session initiator (172.16.57.30:44673) sent
1599 bytes -- responder (172.21.127.218:80) sent 93124 bytes
```

Debugging CBAC

To assist CBAC debugging, you can turn on audit trail messages that will be displayed on the console after each CBAC session closes. Audit trail information is configurable on a per-application basis using the CBAC inspection rules.

To turn on audit trail messages, use the following global configuration command:

Command	Purpose
Router(config)# ip inspect audit-trail	Turn on CBAC audit trail messages.

If required, you can also use the CBAC **debug** commands listed in this section. (Debugging can be turned off for each of the commands in this section by using the **no** form of the command. To disable all debugging, use the privileged EXEC commands **no debug all** or **undebug all**.)

The available **debug** commands are listed in the following categories:

- Generic Debug Commands
- Transport Level Debug Commands
- Application Protocol Debug Commands

For a complete description of the debug commands, refer to the Cisco IOS Release 12.0 *Debug Command Reference*.

Generic Debug Commands

You can use the following generic **debug** commands, entered in privileged EXEC mode:

Command	Purpose
Router# debug ip inspect function-trace	Display messages about software functions called by CBAC.
Router# debug ip inspect object-creation	Display messages about software objects being created by CBAC. Object creation corresponds to the beginning of CBAC-inspected sessions.
Router# debug ip inspect object-deletion	Display messages about software objects being deleted by CBAC. Object deletion corresponds to the closing of CBAC-inspected sessions.
Router# debug ip inspect events	Display messages about CBAC software events, including information about CBAC packet processing.
Router# debug ip inspect timers	Display messages about CBAC timer events such as when a CBAC idle timeout is reached.
Router# debug ip inspect detail	Enable the detailed option, which can be used in combination with other options to get additional information.

Transport Level Debug Commands

You can use the following transport-level **debug** commands, entered in privileged EXEC mode:

Command	Purpose
Router# debug ip inspect tcp	Display messages about CBAC-inspected TCP events, including details about TCP packets.
Router# debug ip inspect udp	Display messages about CBAC-inspected UDP events, including details about UDP packets.

Application Protocol Debug Commands

You can use the following application protocol **debug** command, entered in privileged EXEC mode:

Command	Purpose
Router# debug ip inspect protocol	Display messages about CBAC-inspected protocol events, including details about the protocol's packets. Refer to Table 3 to determine the protocol keyword.

Table 3 identifies application protocol keywords for the **debug ip inspect** command.

Table 3 Application Protocol Keywords for debug ip inspect

Application Protocol	<i>protocol</i> keyword
CU-SeeMe	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the FTP tokens parsed)	ftp-tokens
H.323	h323
HTTP (Java applets)	http

Table 3 Application Protocol Keywords for debug ip inspect (continued)

Application Protocol	<i>protocol</i> keyword
Microsoft NetShow	netshow
UNIX R commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio
RPC	rpc
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Turning Off CBAC

You can turn off CBAC with the **no ip inspect** global configuration command.

Note The **no ip inspect** command removes all CBAC configuration entries and resets all CBAC global timeouts and thresholds to the defaults. All existing sessions are deleted and their associated access lists removed.

In most situations, turning off CBAC has no negative security impact because CBAC creates “permit” access lists. Without CBAC configured, no “permit” access lists are maintained. Therefore, no derived traffic (returning traffic or traffic from the data channels) can go through the firewall. The exception is SMTP and Java blocking. With CBAC turned off, unacceptable SMTP commands or Java applets may go through the firewall.

Configuration Examples

This section provides the following configuration examples:

- Simple CBAC Configuration
- Remote Office to ISP Configuration
- Remote Office to Branch Office Configuration
- Two-interface Branch Office Configuration
- Multiple-interface Branch Office Configuration

The first example develops a CBAC inspection rule and a supporting access control list (ACL), and applies that inspection rule on an ATM interface. This example focuses on how to configure CBAC; it does not provide a complete router configuration and does not describe other elements of the configuration such as the ATM interface.

The remote-office examples also focus on the firewall configuration and do not provide detailed descriptions of other configuration elements such as the Basic Rate Interface (BRI) and dialer interface configurations.

The other examples provide more complete firewall configurations, illustrating ways to apply CBAC in branch office environments with LAN and serial interfaces.

Simple CBAC Configuration

In this example, firewall protection is required against inbound traffic on an ATM interface. This example might apply to sites where local hosts require access to hosts or services on a remote network. The security policy for this site uses access control lists (ACLs) to restrict inbound traffic on the ATM interface to specific ICMP protocol traffic, denying inbound access for TCP and UDP protocol traffic. Inbound access for specific TCP and UDP protocol traffic is provided through dynamic access lists, which are generated according to CBAC inspection rules.

For information on how to select the interface on which to apply CBAC, refer to the “Picking an Interface: Internal or External” section.

Note For Frame Relay or ATM interfaces, you can apply CBAC inspection rules separately on each sub-interface, even though the sub-interfaces are physically connected through one interface.

```

!-----
!Create the Inspection Rule
!-----
!
!Create the CBAC inspection rule "test", allowing inspection of the protocol traffic
!specified by the rule. This inspection rule sets the timeout value to 30 seconds for
!each protocol (except for RPC). The timeout value defines the maximum time that a
!connection for a given protocol can remain active without any traffic passing through
!the router. When these timeouts are reached, the dynamic ACLs that are inserted to
!permit the returning traffic are removed, and subsequent packets (possibly even valid
!ones) are not permitted.
ip inspect name test cuseeme timeout 30
ip inspect name test ftp timeout 30
ip inspect name test h323 timeout 30
ip inspect name test realaudio timeout 30
ip inspect name test rpc program-number 100000
ip inspect name test streamworks timeout 30
ip inspect name test vdolive timeout 30
!
!-----
!Create the Access Control List
!-----
!
!In this example, ACL 105 denies all TCP and UDP protocol traffic. ICMP traffic from
!subnet 192.168.1.0 is permitted to allow access for routing and control traffic.
!ACL 105 specifies that only the return traffic for protocols defined in the inspection
!rule is allow access through the interface where this rule is applied. The final deny
!statement is added for explicitness.
access-list 105 deny TCP any any
access-list 105 deny UDP any any
access-list 105 permit icmp any any echo-reply
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 time-exceeded
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 packet-too-big
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 traceroute
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 unreachable
access-list 105 deny ip any any
!
!-----
!Apply the Inspection Rule and ACL
!-----
!

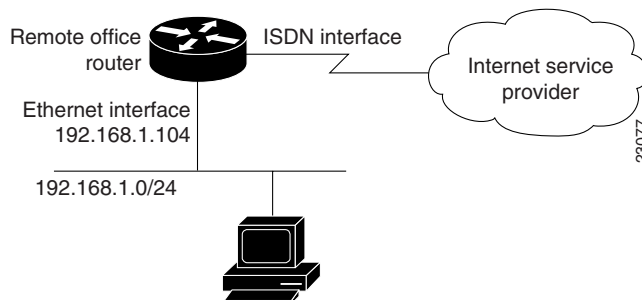
```

```
!In this example, the inspection rule "test" is applied to traffic at interface ATM3/0
!for connections initiated in the outbound direction; that is, from hosts that are
!located on a local network. CBAC creates dynamic access list entries for traffic
!initiated by local hosts. These dynamic entries allow inbound (returning) traffic for
!that connection. ACL 105 is applied at interface ATM3/0 in the inbound direction to
!block traffic initiated from hosts on a remote network that is not part of an existing
!connection.
interface ATM3/0
 ip address 10.1.10.1 255.0.0.0
 ip access-group 105 in
 no ip directed-broadcast
 ip inspect test out
 no shutdown
 atm clock INTERNAL
 atm pvc 7 7 7 aal5snap
 map-group atm
```

Remote Office to ISP Configuration

This example describes one possible Cisco IOS Firewall configuration for a remote office router connected to an Internet service provider (ISP). In this configuration, the site security policy allows hosts on the local network to initiate traffic to the ISP while traffic inbound to the router from the ISP is blocked at the ISDN interface. Specific ICMP control message traffic is permitted through the firewall. No mail or Web services are available from the local network. Figure 4 illustrates this example.

Figure 4 Remote Office to ISP Sample Configuration



The firewall has two interfaces:

- An Ethernet interface connects to the internal protected network.

Interface Ethernet0 has no ACL applied to it, meaning that all traffic initiated on the LAN is allowed access to the ISP. In this configuration example, Network Address Translation (NAT) is not turned on, and the addresses on interface Ethernet0 are reserved IP addresses. In a production environment, addresses on Ethernet0 either must be registered network addresses, or you must turn on NAT to hide these inside addresses from being visible on the Internet.

- An ISDN Basic Rate Interface (BRI) connects the router to the ISP. In this example, a dialer profile is used to control the BRI interface. This means that the ACL and CBAC inspection rules are applied at the dialer interface, not directly at the physical ISDN (BRI) interface using a dialer map.

```

!-----
!General Cisco IOS Firewall Guidelines
!-----
!The following global configuration entries illustrate good security practices.
enable secret 5 <elided>
no ip source-route
no cdp run
!
!-----
!Create the CBAC inspection rule
!-----
!Create the CBAC inspection rule STOP to allow inspection of the protocol traffic
!specified by the rule.
ip inspect name STOP tcp
ip inspect name STOP ftp
ip inspect name STOP smtp
ip inspect name STOP h323
ip inspect name STOP rcmd
!
!-----
!Create Access Control List 105
!-----
!ACL 105 denies all IP protocol traffic except for specific ICMP control traffic.
!This means that only the return traffic for protocols defined in the
!inspection rule and the specified ICMP traffic is allowed access through the
!interface where this rule is applied.
!
!Deny broadcast messages with a source address of 255.255.255.255; this helps to
!prevent broadcast attacks.
access-list 105 deny ip host 255.255.255.255 any
!
!Add anti-spoofing protection by denying traffic with a source address matching a host
!on the Ethernet interface.
acl 105 deny ip 192.168.1.0 0.0.0.255 any
!
!ICMP traffic is not inspected by CBAC. To control the type of ICMP traffic at the
!interface, add static access list entries. This example has the following ICMP
!requirements: outgoing ping commands require echo-reply messages to come back,
!outgoing traceroute commands require time-exceeded messages to come back, path MTU
!discovery requires "too-big" messages to come back, and incoming traceroute
!messages must be allowed. Additionally, permit all "unreachable" messages to come
!back; that is, if a router cannot forward or deliver a datagram, it sends an ICMP
!unreachable
!message back to the source and drops the datagram.
access-list 105 permit icmp any any echo-reply
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 time-exceeded
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 packet-too-big
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 traceroute
access-list 105 permit icmp any 192.168.1.0 0.0.0.255 unreachable
!
!Final deny for explicitness. This entry is not required but helps complete the access
!list picture. By default, the final entry in any access list is an implicit deny of IP
!protocol traffic. This ensures that the firewall blocks any traffic not explicitly
!permitted by the access list.
access-list 105 deny ip any any
!
!-----
!Configure the interface
!-----
!In this example, no ACLs or inspection rules are applied at interface Ethernet0,
!meaning that all traffic on the local network is allowed to go out. This assumes a
!high-level of trust for the users on the local network.
interface Ethernet0
    ip address 192.168.1.104 255.255.255.0
!

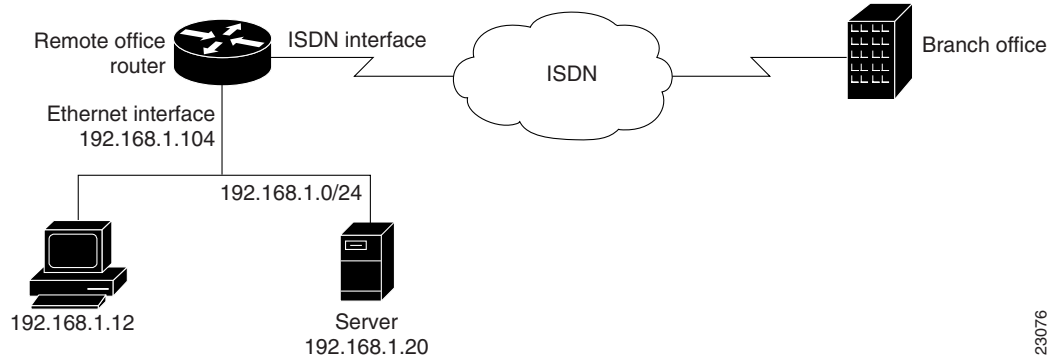
```

```
no ip directed-broadcast
!
!This example uses a dialer profile, so the ACL and CBAC inspection rules are applied
!at the dialer interface, not the physical BRI interface. The dialer pool-member
!command is used to associate the physical interface with a dialer profile.
interface BRI0
  no ip address
  no ip directed-broadcast
  encapsulation ppp
  dialer pool-member 1
  isdn switch-type basic-5ess
!
!-----
!Create the dialer profile.
!-----
!Through the dialer profile, the ACL and CBAC inspection rules are
!applied to every pool member. In this example, the ACL is applied in, meaning that it
!applies to traffic inbound from the ISP. The CBAC inspection rule STOP is applied out,
!meaning that CBAC monitors the traffic through the interface and controls return
!traffic to the router for an existing connection.
interface Dialer0
  ip address negotiated
  ip access-group 105 in
  no ip directed-broadcast
  ip inspect STOP out
  encapsulation ppp
  dialer remote-name <ISP router>
  dialer idle-timeout 500
  dialer string <elided>
  dialer pool 1
  dialer-group 1
  ppp authentication callin
!
!-----
!Additional entries
!-----
!Configure the router to forward packets destined for an unrecognized subnet of
!a directly connected network.
ip classless
!Route traffic to the dialer interface.
ip route 0.0.0.0 0.0.0.0 Dialer0
!Include a dialer list protocol entry to specify the protocol that triggers dialing.
dialer-list 1 protocol ip permit
!Add a user name (name of the router your are configuring) and password for caller
!identification and password authentication with the ISP router.
username <router host name> password 5 <elided>
```

Remote Office to Branch Office Configuration

This example describes one possible Cisco IOS Firewall configuration for a remote office router connected to a branch office. In this configuration, the site security policy allows hosts on the local network to initiate traffic to the branch office. Mail or Web services are available from a server on the local network, and access to these services is available from the branch office. Traffic from the branch office, except for mail and Web traffic, is blocked at the outside interface. Specific ICMP control message traffic is permitted through the firewall. Figure 5 illustrates this example.

Figure 5 Remote Office to Branch Office Sample Configuration



23076

The firewall has two interfaces:

- An Ethernet interface connects to the internal protected network.
Interface Ethernet0 has no ACL applied to it, meaning that all traffic initiated from the LAN is allowed access through the firewall.
- An ISDN Basic Rate Interface (BRI) connects the router to the branch office. In this example, a dialer profile is used to control the BRI interface. This means that the ACL and CBAC inspection rules are applied at dialer interface, not directly at the physical ISDN (BRI) interface.

```

!-----
!General firewall configuration guidelines
!-----
!The following global configuration entries illustrate good security practices.
enable secret 5 <elided>
no ip source-route
no cdp run
!
!-----
!Create the Inspection Rule
!-----
!Create the CBAC inspection rule STOP to allow inspection of the specified protocol
!traffic. Create the inspection rule GO to allow inspection of HTTP and SMTP
!traffic. Note that Java applets will be permitted according to access list 51, which
!is defined later in this sample configuration.
ip inspect name STOP tcp
ip inspect name STOP ftp
ip inspect name STOP smtp
ip inspect name STOP h323
ip inspect name STOP rcmd
ip inspect name GO http java-list 51
ip inspect name GO smtp
!
!-----
!Create Access Control Lists 106 and 51
!-----
!ACL 106 permits mail and Web traffic from any host to the specified server. ACL 106
!denies all other ip protocol traffic except for specific ICMP control traffic.
!This means that only the return traffic for protocols defined in the
!inspection rule and the specified ICMP traffic is allowed access through the
!interface where this rule is applied.
!
!Deny broadcast messages with a source address of 255.255.255.255; this helps to
!prevent broadcast attacks.
access-list 106 deny ip host 255.255.255.255 any
!

```

Remote Office to Branch Office Configuration

```
!Add anti-spoofing protection by denying traffic with a source address matching a host
!on the Ethernet interface.
access-list 106 deny ip 192.168.1.0 0.0.0.255 any
!
!ICMP traffic is not inspected by CBAC. To control the type of ICMP traffic at the
!interface, add static access list entries. This example has the following ICMP
!requirements: outgoing ping commands require echo-reply messages to come back,
!outgoing traceroute commands require time-exceeded messages to come back, path MTU
!discovery requires "too-big" messages to come back, and incoming traceroute must be
!allowed. Additionally, permit all "unreachable" messages to come back; that is, if a
!router cannot forward or deliver a datagram, it sends an ICMP unreachable message back
!to the source and drops the datagram.
access-list 106 permit icmp any any echo-reply
access-list 106 permit icmp any 192.168.1.0 0.0.0.255 time-exceeded
access-list 106 permit icmp any 192.168.1.0 0.0.0.255 packet-too-big
access-list 106 permit icmp any 192.168.1.0 0.0.0.255 traceroute
access-list 106 permit icmp any 192.168.1.0 0.0.0.255 unreachable
!
!Permit mail and Web access to a specific server.
access-list 106 permit tcp any host 192.168.1.20 eq smtp
access-list 106 permit tcp any host 192.168.1.20 eq www
!
!Final deny for explicitness. This entry is not required but helps complete the access
!list picture. By default, the final entry in any access list is an implicit deny of IP
!protocol traffic. This ensures that the firewall blocks any traffic not explicitly
!permitted by the access list.
access-list 106 deny ip any any
!
!Access list 51 defines the sites for Java applet blocking. If the access list denies a
!site, that site is deemed "hostile" and applets from that site are blocked. If the
!access list permits a site, that site is deemed "friendly" and applets from that
!site are not blocked. Java applet blocking is defined in the inspection rule "GO",
!meaning applets are permitted or denied from the sites defined in the access list. In
!this example, access list 51 permits Java applets from any site (source address).
access-list 51 permit any
!
!-----
!Configure the interface.
!-----
!In this example, no ACLs or inspection rules are applied at interface Ethernet0,
!meaning that all traffic on the local network is allowed to go out. This assumes a
!high-level of trust for the users on the local network.
interface Ethernet0
    ip address 192.168.1.104 255.255.255.0
    no ip directed-broadcast
!
!This example uses a dialer profile, so the ACL and CBAC inspection rules are applied
!at the dialer interface, not the physical BRI interface. The dialer pool-member
!command is used to associate the physical interface with a dialer profile.
interface BRI0
    no ip address
    no ip directed-broadcast
    encapsulation ppp
    dialer pool-member 1
    isdn switch-type basic-5ess
!
!-----
!Apply the ACL and CBAC inspection rules at the dialer interface.
!-----
!Through the dialer profile, the ACL and CBAC inspection rules are
!applied to every pool member. In this example, the ACL is applied in, meaning that it
!applies to traffic inbound from the branch office. The CBAC inspection rule STOP is
!applied out, meaning that CBAC monitors the traffic and controls return traffic to the
!router for an existing connection. The CBAC inspection rule GO is applied in,
!protecting against certain types of DoS attacks as described in this document. Note
```

```

!that the GO inspection rule does not control return traffic because there is no ACL
!blocking traffic in that direction; however, it does monitor the connections.
interface Dialer0
  ip address <ISDN interface address>
  ip access-group 106 in
  no ip directed-broadcast
  ip inspect STOP out
  ip inspect GO in
  encapsulation ppp
  dialer remote-name <branch office router>
  dialer idle-timeout 500
  dialer string <elided>
  dialer pool 1
  dialer-group 1
  ppp authentication
!
!-----
Additional entries
!-----
!Configure the router to forward packets destined for an unrecognized subnet of
!a directly connected network.
ip classless
!Route traffic to the dialer interface.
ip route 0.0.0.0 0.0.0.0 Dialer0
!Include a dialer list protocol entry to specify the protocol that triggers dialing.
dialer-list 1 protocol ip permit
!Add a user name (name of the router your are configuring) and password for caller
!identification and password authentication with the ISP router.
username <router host name> password 5 <elided>

```

Two-interface Branch Office Configuration

This sample configuration file describes a firewall configured with CBAC. The firewall is positioned between a protected field office's internal network and a WAN connection to the corporate headquarters. CBAC is configured on the firewall in order to protect the internal network from potential network threats coming from the WAN side.

The firewall has two interfaces configured:

- Interface Ethernet0 connects to the internal protected network
- Interface Serial0 connects to the WAN with Frame Relay

```

!-----
! This first section contains some configuration that is not required for CBAC,
! but illustrates good security practices. Note that there are no
! services on the Ethernet side. Email is picked up via POP from a server on the
! corporate side.
!-----
!
hostname user1-examplecorp-fr
!
boot system flash c1600-fw1600-1
enable secret 5 <elided>
!
username user1 password <elided>
ip subnet-zero
no ip source-route
ip domain-name example.com
ip name-server 172.19.2.132
ip name-server 198.92.30.32
!
!

```

Two-interface Branch Office Configuration

```
!-----
!The next section includes configuration required specifically for CBAC
!-----
!
!The following commands define the inspection rule "myfw", allowing
!the specified protocols to be inspected. Note that Java applets will be permitted
!according to access list 51, defined later in this configuration.
ip inspect name myfw cuseeme timeout 3600
ip inspect name myfw ftp timeout 3600
ip inspect name myfw http java-list 51 timeout 3600
ip inspect name myfw rcmd timeout 3600
ip inspect name myfw realaudio timeout 3600
ip inspect name myfw smtp timeout 3600
ip inspect name myfw tftp timeout 30
ip inspect name myfw udp timeout 15
ip inspect name myfw tcp timeout 3600
!
!The following interface configuration applies the "myfw" inspection rule to
!inbound traffic at Ethernet 0. Since this interface is on the internal network
!side of the firewall, traffic entering Ethernet 0 is actually
!exiting the internal network. Applying the inspection rule to this interface causes
!inbound traffic (which is exiting the network) to be inspected; return traffic will
!only be permitted back through the firewall if part of a session which began from
!within the network.
!Also note that access list 101 is applied to inbound traffic at Ethernet 0.
!Any traffic that passes the access list will be inspected by CBAC.
!(Traffic blocked by the access list will not be inspected.)
interface Ethernet0
    description ExampleCorp Ethernet chez user1
    ip address 172.19.139.1 255.255.255.248
    ip broadcast-address 172.19.131.7
    no ip directed-broadcast
    no ip proxy-arp
    ip inspect myfw in
    ip access-group 101 in
    no cdp enable
!
interface Serial0
    description Frame Relay (Telco ID 22RTQQ062438-001) to ExampleCorp HQ
    no ip address
    ip broadcast-address 0.0.0.0
    encapsulation frame-relay IETF
    no arp frame-relay
    bandwidth 56
    service-module 56k clock source line
    service-module 56k network-type dds
    frame-relay lmi-type ansi
!
!Note that the following interface configuration applies access list 111 to
!inbound traffic at the external serial interface. (Inbound traffic is
!entering the network.) When CBAC inspection occurs on traffic exiting the
!network, temporary openings will be added to access list 111 to allow returning
!traffic that is part of existing sessions.
!
interface Serial0.1 point-to-point
    ip unnumbered Ethernet0
    ip access-group 111 in
    bandwidth 56
    no cdp enable
    frame-relay interface-dlci 16
!
ip classless
ip route 0.0.0.0 0.0.0.0 Serial0.1
!
!The following access list defines "friendly" and "hostile" sites for Java
```

```

!applet blocking. Because Java applet blocking is defined in the inspection
!rule "myfw" and references access list 51, applets will be actively denied
!if they are from any of the "deny" addresses and allowed only if they are from
!either of the two "permit" networks.
!
access-list 51 deny 172.19.1.203
access-list 51 deny 172.19.2.147
access-list 51 permit 172.18.0.0 0.1.255.255
access-list 51 permit 192.168.1.0 0.0.0.255
access-list 51 deny any
!
!The following access list 101 is applied to interface Ethernet 0 above.
!This access list permits all traffic that should be CBAC inspected, and also
!provides anti-spoofing. The access list is deliberately set up to deny unknown
!IP protocols, because no such unknown protocols will be in legitimate use.
!
access-list 101 permit tcp 172.19.139.0 0.0.0.7 any
access-list 101 permit udp 172.19.139.0 0.0.0.7 any
access-list 101 permit icmp 172.19.139.0 0.0.0.7 any
access-list 101 deny ip any any
!
!The following access list 111 is applied to interface Serial 0.1 above.
!This access list filters traffic coming in from the external side. When
!CBAC inspection occurs, temporary openings will be added to the beginning of
!this access list to allow return traffic back into the internal network.
!This access list should restrict traffic that will be inspected by
!CBAC. (Remember that CBAC will open holes as necessary to permit returning traffic.)
!Comments precede each access list entry. These entries are not all specifically
!related to CBAC, but are created to provide general good security.
!
!Anti-spoofing.
access-list 111 deny ip 172.19.139.0 0.0.0.7 any
!Sometimes EIGRP is run on the Frame Relay link. When you use an
!input access list, you have to explicitly allow even control traffic.
!This could be more restrictive, but there would have to be entries
!for the EIGRP multicast as well as for the office's own unicast address.
access-list 111 permit igrp any any
!
!These are the ICMP types actually used...
!administratively-prohibited is useful when you are trying to figure out why
!you cannot reach something you think you should be able to reach.
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 administratively-prohibited
!
!This allows network admins at headquarters to ping hosts at the field office:
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 echo
!
!This allows the field office to do outgoing pings
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 echo-reply
!
!Path MTU discovery requires too-big messages
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 packet-too-big
!
!Outgoing traceroute requires time-exceeded messages to come back
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 time-exceeded
!
!Incoming traceroute
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 traceroute
!
!Permits all unreachable because if you are trying to debug
!things from the remote office, you want to see them. If nobody ever did
!any debugging from the network, it would be more appropriate to permit only
!port unreachable or no unreachable at all.
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 unreachable
!
!

```

Multiple-interface Branch Office Configuration

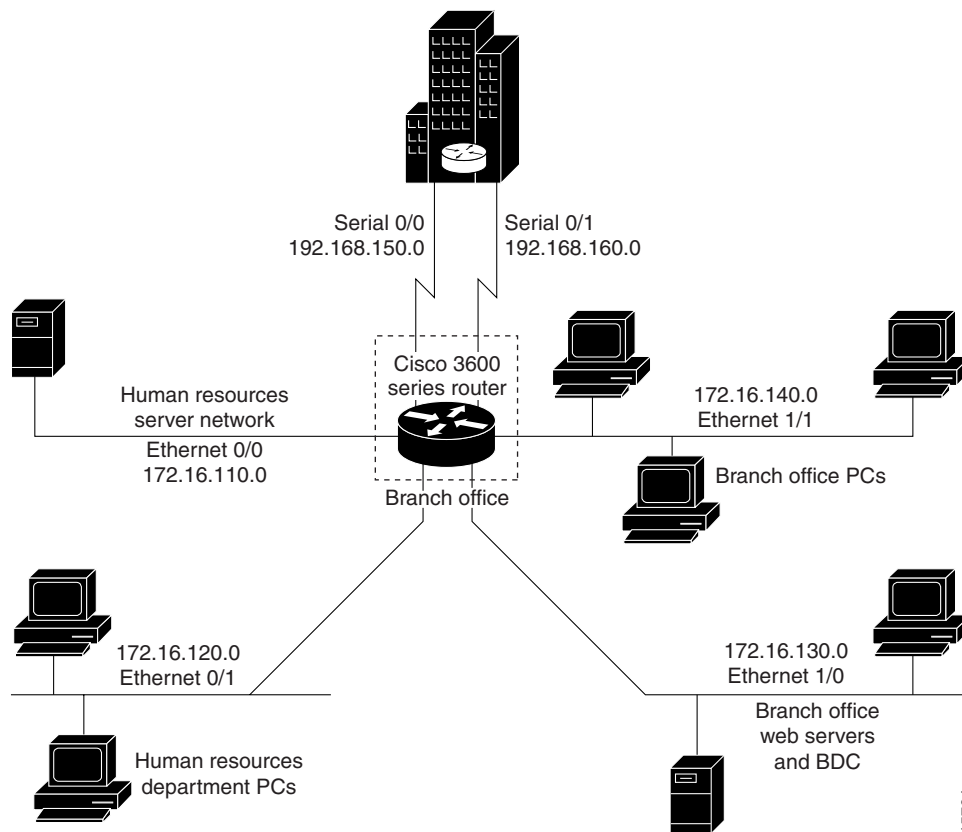
```
!These next two entries permit users on most ExampleCorp networks to Telnet to
!a host in the field office. This is for remote administration by the network admins.
access-list 111 permit tcp 172.18.0.0 0.1.255.255 host 172.19.139.1 eq telnet
access-list 111 permit tcp 192.168.1.0 0.0.0.255 host 172.19.139.1 eq telnet
!
!Final deny for explicitness
access-list 111 deny ip any any
!
no cdp run
snmp-server community <elided> RO
!
line con 0
  exec-timeout 0 0
  password <elided>
  login local
line vty 0
  exec-timeout 0 0
  password <elided>
  login local
  length 35
line vty 1
  exec-timeout 0 0
  password 7 <elided>
  login local
line vty 2
  exec-timeout 0 0
  password 7 <elided>
  login local
line vty 3
  exec-timeout 0 0
  password 7 <elided>
  login local
line vty 4
  exec-timeout 0 0
  password 7 <elided>
  login local
!
scheduler interval 500
end
```

Multiple-interface Branch Office Configuration

In this configuration example, a single Cisco 3600 series firewall router is positioned at a branch office. It has four internal networks and two WAN connections to the corporate headquarters. CBAC is configured on the firewall to protect two of the internal networks from potential network threats coming from the WAN side and from less secure internal networks. Anti-spoofing protection is added at each interface with client systems. Figure 6 illustrates this configuration.

Note This example shows a moderately high level of trust by the administrators toward the expected users. Additional protection could be added to this configuration for a situation in a lower level of trust. That configuration would include ICMP filtering statements, significantly more protocol and address control through the use of more restrictive access control lists, and anti-spoofing applied everywhere. This configuration does not contain those additional restrictions because that would detract from the CBAC example.

Figure 6 Sample Cisco IOS Firewall Application Environment



The branch office has this sample network configuration:

- Ethernet interface 0/0 supports the Human Resources department servers. This network includes an email (SMTP and POP3) host and a Windows NT server. The Windows NT server is the Primary Domain Controller (PDC) for the Human Resources domain and has a trust relationship with the rest of the company; however, it contains applications and databases that must not be accessed by the rest of the company or the other groups in the branch office. The devices on this LAN are accessible only by users in the Human Resources department on Ethernet interface 0/1. The Mail server must be able to send and receive email (through SMTP sessions) with all other devices. The Windows 95 machines can use this machine as their email server (for sending email through SMTP sessions) and as a repository for accumulating email that they can then download through POP3 sessions. No one else in the company is allowed to form POP3 sessions to any machine on this LAN.
- Ethernet interface 0/1 supports the Windows 95 computers in the Human Resources department. These users must have access to the Human Resources mail servers located on Ethernet interface 0/0 as well as access to the rest of the company. Access to the Windows NT server resources are controlled through the Windows NT permissions assigned to each user in the Windows NT domain.
- Ethernet interface 1/0 supports the branch office web servers, which can be accessed by everyone in the company. These servers use TCP ports 80 (HTTP) and 443 (SHTTP) for inbound Web access. This network also includes a backup domain controller (BDC) for the overall domain that is also used as file, print, and service server.

Ethernet interface 1/1 supports all users who are not in the Human Resources department. These users have no access to the Human Resources department servers, but they can access the other network interfaces and the serial interfaces for WAN connectivity. Serial interface 0/0 and 0/1 connect to the WAN with T1 links (links to corporate headquarters). In this sample configuration, the Domain Name System (DNS) servers are located somewhere within the rest of the company.

Additionally, network management (SNMP) and Telnet sessions are limited to the management network (192.168.55.0), which is located somewhere within the rest of the company across the serial interface.

```
! -----
! This first section contains some configuration that is not required
! for CBAC, but illustrates good security practices.
! -----
!Add this line to get timestamps on the syslog messages.
service timestamps log datetime localtime show-timezone
!
hostname Router1
!
boot system flash c3600-fw3600-1
!
! Configure AAA user authentication.
aaa new-model
aaa authentication login lista tacacs+ enable
!
enable secret 5 <elided>
ip subnet-zero
!
! Disable source routing to help prevent spoofing.
no ip source-route
!
! Set up the domain name and server IP addresses.
ip domain-name example.com
ip name-server 192.168.55.132
ip name-server 192.168.27.32
!
! The audit-trail command enables the delivery of specific CBAC messages
! through the syslog notification process.
ip inspect audit-trail
!
! Establish the time-out values for DNS queries. When this idle-timer expires,
! the dynamic ACL entries that were created to permit the reply to a DNS request
! will be removed and any subsequent packets will be denied.
ip inspect dns-timeout 10
!
!-----
!The next section includes configuration statements required
!specifically for CBAC.
!-----
! Define the CBAC inspection rule "inspect1", allowing the specified protocols to be
! inspected. The first rule enables SMTP specific inspection. SMTP inspection causes
! the exchange of the SMTP session to be inspected for illegal commands. Any packets
! with illegal commands are dropped, and the SMTP session will hang and eventually
! time out.
ip inspect name inspect1 smtp timeout 300
!
! In the next two lines of inspect1, define the maximum time that each of the UDP and
! TCP sessions are allowed to continue without any traffic passing
! through the router. When these timeouts are reached, the dynamic ACLs that
! are inserted to permit the returning traffic are removed and subsequent packets
! (possibly even valid ones) will not be permitted.
ip inspect name inspect1 udp timeout 300
ip inspect name inspect1 tcp timeout 300
!
```

```

! Define the CBAC inspection rule "inspect2", allowing the specified protocols to be
! inspected. These rules are similar to those used in the inspection rule "inspect1,"
! except that on the interfaces where this rule is applied, SMTP sessions are not
! expected to go through; therefore, the SMTP rule element is not applied here.
ip inspect name inspect2 udp timeout 300
ip inspect name inspect2 tcp timeout 3600
!
!-----
! The next section shows the Ethernet interface configuration statements for each
! interface, including access lists and inspections rules.
!-----
! Apply the "inspect1" inspection rule to sessions that are initiated in the outbound
! direction (toward the LAN) at Ethernet interface 0/0. All packets in these sessions
! will be inspected by CBAC. Provided that network traffic passes the Access Control
! List (ACL) restrictions, traffic is then inspected by CBAC for access through the
! Cisco IOS Firewall. Traffic blocked by the access list is not inspected by CBAC.
! Access list 110 is applied to outbound traffic on this interface.
interface Ethernet0/0
  description HR_Server Ethernet
  ip address 172.16.110.1 255.255.255.0
  ip access-group 110 out
  no ip directed-broadcast
  no ip proxy-arp
  ip inspect inspect1 out
  no cdp enable
!
! Apply access list 120 to inbound traffic on Ethernet interface 0/1.
! Applying access list 120 to inbound traffic provides anti-spoofing on this interface
! by dropping traffic with a source address matching the IP address on a network other
! than Ethernet 0/1. The IP helper address lists the IP address of the DHCP server on
! Ethernet interface 1/0.
interface Ethernet0/1
  description HR_client Ethernet
  ip address 172.16.120.1 255.255.255.0
  ip access-group 120 in
  ip helper-address 172.16.130.66
  no ip directed-broadcast
  no ip proxy-arp
  no cdp enable
!
! Apply the "inspect2" inspection rule to sessions that are initiated in the outbound
! direction (toward the LAN) at Ethernet interface 1/0. Provided that network traffic
! passes the Access Control List (ACL) restrictions, traffic is then inspected by CBAC
! through the Cisco IOS Firewall. Traffic blocked by the access list is not inspected
! by CBAC. Access list 130 is applied to outbound traffic on this interface.
interface Ethernet1/0
  description Web_server Ethernet
  ip address 172.16.130.1 255.255.255.0
  ip access-group 130 out
  no ip directed-broadcast
  no ip proxy-arp
  ip inspect inspect2 out
  no cdp enable
!
! Apply access list 140 to inbound traffic at Ethernet interface 1/1. This
! provides anti-spoofing on the interface by dropping traffic with a source address
! matching the IP address of a network other than Ethernet 1/1. The IP helper address
! lists the IP address of the DHCP server on Ethernet interface 1/0.
interface Ethernet1/1
  description Everyone_else Ethernet
  ip address 172.16.140.1 255.255.255.0
  ip access-group 140 in
  ip helper-address 172.16.130.66
  no ip directed-broadcast

```

Multiple-interface Branch Office Configuration

```
no ip proxy-arp
no cdp enable
!
!-----
! The next section configures the serial interfaces, including access lists.
!-----
! Apply access list 150 to Serial interfaces 0/0. This provides anti-spoofing on the
! serial interface by dropping traffic with a source address matching the IP address
! of a host on Ethernet interface 0/0, 0/1, 1/0, or 1/1.
interface Serial0/0
  description T1 to HQ
  ip address 192.168.150.1 255.255.255.0
  ip access-group 150 in
  bandwidth 1544
!
interface Serial1/1
  description T1 to HQ
  ip address 192.168.160.1 255.255.255.0
  ip access-group 150 in
  bandwidth 1544
!
! -----
! Configure routing information.
! -----
router igrp 109
network 172.16.0.0
network 192.168.150.0
network 192.168.160.0
!
! Define protocol forwarding on the firewall. When you turn on a related command,
! ip helper-address, you forward every IP broadcast in the ip forward protocol
! command list, including several which are on by default: TFTP (port 69),
! DNS (port 53), Time service (port 37), NetBIOS Name Server (port 137),
! NetBIOS Datagram Server (port 138), BOOTP client and server datagrams
! (ports 67 and 68), and TACACS service (port 49). One common
! application that requires helper addresses is Dynamic Host Configuration
! Protocol (DHCP). DHCP information is carried inside of BOOTP packets. The
! "no ip forward protocol" statements turn off forwarding for the specified protocols.
no ip forward-protocol udp netbios-ns
no ip forward-protocol udp netbios-dgm
no ip forward-protocol udp tacacs
no ip forward-protocol udp tftp
ip forward-protocol udp bootpc
!
! Add this line to establish where router SYSLOG messages are sent. This includes the
! CBAC messages.
logging 192.168.55.131
!
! -----
! Define the configuration of each access list.
! -----
! Defines Telnet controls in access list 12.
access-list 12 permit 192.168.55.0 0.0.0.255
!
! Defines SNMP controls in access list 13.
access-list 13 permit 192.168.55.12
access-list 13 permit 192.168.55.19
!
! Access list 110 permits TCP and UDP protocol traffic for
! specific ports and with a source address on Ethernet interface 0/1. The access list
! denies IP protocol traffic with any other source and destination address. The
! access list permits ICMP access for any source and destination
! address. Access list 110 is deliberately set up to deny unknown IP protocols
! because no such unknown protocols will be in legitimate use. Access list
! 110 is applied to outbound traffic at Ethernet interface 0/0. In ACL 110,
```

```

! network traffic is being allowed access to the ports on any server on the HR server
! network. In less trusted environments, this can be a security problem; however, you
! can limit access more severely by specifying specific destination addresses in the
! ACL statements.
access-list 110 permit tcp 172.16.120.0 0.0.0.255 any eq smtp
access-list 110 permit tcp 172.16.120.0 0.0.0.255 any eq pop3
access-list 110 permit tcp 172.16.120.0 0.0.0.255 any eq 110
access-list 110 permit udp any any eq 137
access-list 110 permit udp any any eq 138
access-list 110 permit udp any any eq 139
access-list 110 permit icmp any any
access-list 110 deny ip any any!
!
! Access-list 120 permits TCP, UDP, and ICMP protocol traffic with a source address
! on Ethernet interface 0/1, but denies all other IP protocol traffic. Access list
! 120 is applied to inbound traffic on Ethernet interface 0/1.
access-list 120 permit tcp 172.16.120.0 0.0.0.255 any
access-list 120 permit udp 172.16.120.0 0.0.0.255 any
access-list 120 permit icmp 172.16.120.0 0.0.0.255 any
access-list 120 deny ip any any
!
! Access list 130 permits TCP, UDP, and ICMP protocol traffic for specific ports and
! with any source and destination address. It opens access to the web server and to
! all NBT services to the rest of the company, which can be controlled through the
! trust relations on the Windows NT servers. The bootpc entry permits access to the
! DHCP server. Access list 130 denies all other IP protocol traffic. Access list 130 is
! applied to outbound traffic at Ethernet interface 1/0.
access-list 130 permit tcp any any eq www
access-list 130 permit tcp any any eq 443
access-list 130 permit tcp any any eq 110
access-list 130 permit udp any any eq 137
access-list 130 permit udp any any eq 138
access-list 130 permit udp any any eq 139
access-list 130 permit udp any any eq bootpc
access-list 130 permit icmp any any
access-list 130 deny ip any any
!
! Access list 140 permits TCP, UDP, and ICMP protocol traffic with a source address on
! Ethernet interface 1/1, and it denies all other IP protocol traffic. Access list 140
! is applied to inbound traffic at Ethernet interface 1/1.
access-list 140 permit tcp 172.16.140.0 0.0.0.255 any
access-list 140 permit udp 172.16.140.0 0.0.0.255 any
access-list 140 permit icmp 172.16.140.0 0.0.0.255 any
access-list 140 deny ip any any
!
! Access list 150 denies IP protocol traffic with a source address on Ethernet
! interfaces 0/0, 0/1, 1/0, and 1/1, and it permits IP protocol traffic with any other
! source and destination address. Access list 150 is applied to inbound traffic
! on each of the serial interfaces.
access-list 150 deny ip 172.16.110.0 0.0.0.255 any
access-list 150 deny ip 172.16.120.0 0.0.0.255 any
access-list 150 deny ip 172.16.130.0 0.0.0.255 any
access-list 150 deny ip 172.16.140.0 0.0.0.255 any
access-list 150 permit ip any any
!
! Disable Cisco Discovery Protocol.
no cdp run
!
snmp-server community <elided> ro 13
tacacs-server host 192.168.55.2
tacacs-server key <elided>
!
! -----
! Configures the router console port and the virtual terminal line interfaces,
! including AAA authentication at login. Authentication is required for users defined

```

```
! in "lista." Access-class 12 is applied on each line, restricting Telnet access to
! connections with a source address on the network management network.
! -----
line console 0
exec-timeout 3 00
login authentication lista
line aux 0
exec-timeout 3 00
login authentication lista
line vty 0
    exec-timeout 1 30
    login authentication lista
    access-class 12 in
line vty 1
    exec-timeout 1 30
    login authentication lista
    access-class 12 in
line vty 2
    exec-timeout 1 30
    login authentication lista
    access-class 12 in
line vty 3
    exec-timeout 1 30
    login authentication lista
    access-class 12 in
line vty 4
    exec-timeout 1 30
    login authentication lista
    access-class 12 in
!
end
```

Command Reference

This section documents new or modified commands. All other commands used with this feature are documented in the Cisco IOS Release 12.0(5)T command reference publications.

- **ip inspect name (global configuration)**
- **ip inspect alert-off**

In Cisco IOS Release 12.0(1)T or later, you can search and filter the output for **show** and **more** commands. This functionality is useful when you need to sort through large amounts of output, or if you want to exclude output that you do not need to see.

To use this functionality, enter a **show** or **more** command followed by the “pipe” character (**|**), one of the keywords **begin**, **include**, or **exclude**, and an expression that you want to search or filter on:

```
command | {begin | include | exclude} regular-expression
```

Following is an example of the **show atm vc** command in which you want the command output to begin with the first line where the expression “PeakRate” appears:

```
show atm vc | begin PeakRate
```

For more information on the search and filter functionality, refer to the Cisco IOS Release 12.0(1)T feature module titled *CLI String Search*.

ip inspect name (global configuration)

To define a set of inspection rules, use the **ip inspect name** global configuration command. Use the **no** form of this command to remove the inspection rule for a protocol or to remove the entire set of inspection rules.

ip inspect name *inspection-name* *protocol* [**alert** {**on** | **off**}] [**audit-trail** {**on** | **off**}] [**timeout** *seconds*]

or

ip inspect name *inspection-name* **http** [**java-list** *access-list*] [**alert** {**on** | **off**}] [**audit-trail** {**on** | **off**}] [**timeout** *seconds*] (Java protocol only)

or

ip inspect name *inspection-name* **rpc** *program-number* *number* [**wait-time** *minutes*] [**alert** {**on** | **off**}] [**audit-trail** {**on** | **off**}] [**timeout** *seconds*] (RPC protocol only)

or

ip inspect name *inspection-name* **fragment** [**max** *number* **timeout** *seconds*]

no ip inspect name *inspection-name* *protocol* (removes the inspection rule for a protocol)

no ip inspect name *inspection-name* **fragment** (removes fragment inspection for a rule)

no ip inspect name (removes the entire set of inspection rules)

Syntax Description

<i>inspection-name</i>	Names the set of inspection rules. If you want to add a protocol to an existing set of rules, use the same <i>inspection-name</i> as the existing set of rules.
<i>protocol</i>	A protocol keyword listed in Table 4.
alert { on off }	(Optional) For each inspected protocol, the generation of alert messages can be set be on or off . If no option is selected, alerts are generated based on the setting of the ip inspect alert-off command.
audit-trail { on off }	(Optional) For each inspected protocol, audit trail can be set on or off . If no option is selected, audit trail message are generated based on the setting of the ip inspect audit-trail command.
http	(Optional) Specifies the HTTP protocol for Java applet blocking.
timeout <i>seconds</i>	(Optional) To override the global TCP or UDP idle timeouts for the specified protocol, specify the number of seconds for a different idle timeout. This timeout overrides the global TCP and UPD timeouts but will not override the global DNS timeout.

- java-list** *access-list* (Optional) Specifies the access list (name or number) to use to determine “friendly” sites. This keyword is available only for the HTTP protocol, for Java applet blocking. Java blocking only works with standard access lists.
- rpc program-number** *number* Specifies the program number to permit. This keyword is available only for the RPC protocol.
- wait-time** *minutes* (Optional) Specifies the number of minutes to keep a small hole in the firewall to allow subsequent connections from the same source address and to the same destination address and port. The default wait-time is zero minutes. This keyword is available only for the RPC protocol.
- fragment** Specifies fragment inspection for the named rule.
- max** *number* Specifies the maximum number of unassembled packets for which state information (structures) is allocated by Cisco IOS software. Unassembled packets are packets that arrive at the router interface before the initial packet for a session. The acceptable range is 50 through 10000. The default is 256 state entries.

Memory is allocated for the state structures, and setting this value to a larger number may cause memory resources to be exhausted.
- timeout** *seconds* (fragmentation) Configures the number of seconds that a packet state structure remains active. When the timeout value expires, the router drops the unassembled packet, freeing that structure for use by another packet. The default timeout value is one second.

If this number is set to a value greater than one second, it will be automatically adjusted by the Cisco IOS software when the number of free state structures goes below certain thresholds: when the number of free states is less than 32, the timeout will be divided by 2. When the number of free states is less than 16, the timeout will be set to 1 second.

Table 4 Protocol Keywords

Protocol	<i>protocol</i> Keyword
Transport-Layer Protocols	
TCP	tcp
UDP	udp
Application-Layer Protocols	
CU-SeeMe	cuseeme
FTP	ftp
Java	http
H.323	h323
Microsoft NetShow	netshow

Table 4 Protocol Keywords (continued)

Protocol	<i>protocol</i> Keyword
UNIX R commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio
RPC	rpc
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Defaults

No inspection rules are defined until you define them using this command.

Command Modes

Global configuration

Command History

Release	Modification
11.2P	This command was introduced.
12.0(5)T	Introduced configurable alert and audit trail, IP fragmentation checking, and NetShow protocol support.

Usage Guidelines

To define a set of inspection rules, enter this command for each protocol that you want Context-based Access Control (CBAC) to inspect, using the same *inspection-name*. Give each set of inspection rules a unique *inspection-name*. Define either one or two sets of rules per interface—you can define one set to examine both inbound and outbound traffic; or you can define two sets: one for outbound traffic and one for inbound traffic.

To define a single set of inspection rules, configure inspection for all the desired application-layer protocols, and for TCP or UDP as desired. This combination of TCP, UDP, and application-layer protocols join together to form a single set of inspection rules with a unique name.

In general, when inspection is configured for a protocol, return traffic entering the internal network will be permitted only if the packets are part of a valid, existing session for which state information is being maintained.

TCP and UDP Inspection

You can configure TCP and UDP inspection to permit TCP and UDP packets to enter the internal network through the firewall, even if the application-layer protocol is not configured to be inspected. However, TCP and UDP inspection do not recognize application-specific commands, and therefore might not permit all return packets for an application, particularly if the return packets have a different port number than the previous exiting packet.

Any application-layer protocol that is inspected will take precedence over the TCP or UDP packet inspection. For example, if inspection is configured for FTP, all control channel information will be recorded in the state table, and all FTP traffic will be permitted back through the firewall if the control channel information is valid for the state of the FTP session. The fact that TCP inspection is configured is irrelevant.

With TCP and UDP inspection, packets entering the network must exactly match an existing session: the entering packets must have the same source/destination addresses and source/destination port numbers as the exiting packet (but reversed). Otherwise, the entering packets will be blocked at the interface.

Application-Layer Protocol Inspection

In general, if you configure inspection for an application-layer protocol, packets for that protocol should be permitted to exit the firewall (by configuring the correct ACL), and packets for that protocol will only be allowed back in through the firewall if they belong to a valid existing session. Each protocol packet is inspected to maintain information about the session state.

Java, H.323, RPC, and SMTP, and SQL*Net inspection have additional information, described in the next four sections.

Java Inspection

Java inspection enables Java applet filtering at the firewall. Java applet filtering distinguishes between trusted and untrusted applets by relying on a list of external sites that you designate as “friendly.” If an applet is from a friendly site, the firewall allows the applet through. If the applet is not from a friendly site, the applet will be blocked. Alternately, you could permit applets from all sites except for sites specifically designated as “hostile.”

Note Before you configure Java inspection, you must configure a standard access list that defines “friendly” and “hostile” external sites. You configure this access list to permit traffic from friendly sites, and to deny traffic from hostile sites. If you do not configure an access list, but use a “placeholder” access list in the **ip inspect name inspection-name http** command, all Java applets will be blocked.



Caution CBAC does not detect or block encapsulated Java applets. Therefore, Java applets that are wrapped or encapsulated, such as applets in .zip or .jar format, are *not* blocked at the firewall. CBAC also does not detect or block applets loaded via FTP, gopher, or HTTP on a nonstandard port.

H.323 Inspection

If you want CBAC inspection to work with NetMeeting 2.0 traffic (an H.323 application-layer protocol), you must also configure inspection for TCP, as described in the chapter “Configuring Context-Based Access Control” in the Cisco IOS Release 12.0 *Security Configuration Guide*. This requirement exists because NetMeeting 2.0 uses an additional TCP channel not defined in the H.323 specification.

RPC Inspection

RPC inspection allows the specification of various program numbers. You can define multiple program numbers by creating multiple entries for RPC inspection, each with a different program number. If a program number is specified, all traffic for that program number will be permitted. If a program number is not specified, all traffic for that program number will be blocked. For example, if you created an RPC entry with the NFS program number, all NFS traffic will be allowed through the firewall.

SMTP Inspection

SMTP inspection causes SMTP commands to be inspected for illegal commands. Any packets with illegal commands are dropped, and the SMTP session will hang and eventually time out. An illegal command is any command except for the following legal commands:

- DATA
- EHLO
- EXPN
- HELO
- HELP
- MAIL
- NOOP
- QUIT
- RCPT
- RSET
- SAML
- SEND
- SOML
- VRFY

Use of the **timeout** Keyword

If you specify a timeout for any of the transport-layer or application-layer protocols, the timeout will override the global idle timeout for the interface that the set of inspection rules is applied to.

If the protocol is TCP or a TCP application-layer protocol, the timeout will override the global TCP idle timeout. If the protocol is UDP or a UDP application-layer protocol, the timeout will override the global UDP idle timeout.

If you do not specify a timeout for a protocol, the timeout value applied to a new session of that protocol will be taken from the corresponding TCP or UDP global timeout value valid at the time of session creation.

IP Fragmentation Inspection

CBAC inspection rules can help protect hosts against certain denial-of-service attacks involving fragmented IP packets. Even though the firewall keeps an attacker from making actual connections to a given host, the attacker may still be able to disrupt services provided by that host. This is done by sending many non-initial IP fragments or by sending complete fragmented packets through a router with an ACL that filters the first fragment of a fragmented packet. These fragments can tie up resources on the target host as it tries to reassemble the incomplete packets.

Using fragmentation inspection, the firewall maintains an *interfragment state* (structure) for IP traffic. Non-initial fragments are discarded unless the corresponding initial fragment was permitted to pass through the firewall. Non-initial fragments received before the corresponding initial fragments are discarded.

Note Fragmentation inspection can have undesirable effects in certain cases, because it can result in the firewall discarding any packet whose fragments arrive out of order. There are many circumstances that can cause out-of-order delivery of legitimate fragments. Apply fragmentation inspection in situations where legitimate fragments, which are likely to arrive out of order, might have a severe performance impact.

Because routers running Cisco IOS software are used in a very large variety of networks, and because the CBAC feature is often used to isolate parts of internal networks from one another, the fragmentation inspection feature is not enabled by default. Fragmentation detection must be explicitly enabled for an inspection rule using the **ip inspect name** command. Unfragmented traffic is never discarded because it lacks a fragment state. Even when the system is under heavy attack with fragmented packets, legitimate fragmented traffic, if any, will still get some fraction of the firewall's fragment state resources, and legitimate, unfragmented traffic can flow through the firewall unimpeded.

Examples

The following example causes the software to inspect TCP sessions and UDP sessions, and to specifically allow CU-SeeMe, FTP, and RPC traffic back through the firewall for existing sessions only. For UDP traffic, audit-trail is on. For FTP traffic, the idle timeout is set to override the global TCP idle timeout. For RPC traffic, program numbers 100003, 100005, and 100021 are permitted:

```
ip inspect name myrules tcp
ip inspect name myrules udp audit-trail on
ip inspect name myrules cuseeme
ip inspect name myrules ftp timeout 120
ip inspect name myrules rpc program-number 100003
ip inspect name myrules rpc program-number 100005
ip inspect name myrules rpc program-number 100021
```

The following example adds fragment checking to software inspection of TCP and UDP sessions for the rule named *myname*. In this example, the firewall software will allocate 100 state structures, and the timeout value for dropping unassembled packets is set to 4 seconds. If 100 initial fragments for 100 different packets are sent through the router, all of the state structures will be used up. The initial fragment for packet 101 will be dropped. Additionally, if the number of free state structures (structures available for use by unassembled packets) drops below the threshold values, 32 or 16, the timeout value is automatically reduced to 2 or 1, respectively. Changing the timeout value frees up packet state structures more quickly:

```
ip inspect name myrules tcp
ip inspect name myrules udp audit-trail on
ip inspect name myrules cuseeme
ip inspect name myrules ftp timeout 120
ip inspect name myrules rpc program-number 100003
ip inspect name myrules rpc program-number 100005
ip inspect name myrules rpc program-number 100021
ip inspect name myrules fragment max 100 timeout 4
```

Related Commands

Command	Description
ip inspect (interface configuration)	Applies a set of inspection rules to an interface.
ip inspect audit-trail	Turns on CBAC audit trail messages, which will be displayed on the console after each CBAC session closes.
ip inspect alert-off	Disables CBAC alert messages, which are displayed on the console.

ip inspect alert-off

To disable CBAC alert messages, which are displayed on the console, use the **ip inspect alert off** global configuration command. To enable CBAC alert messages, use the **no** form of this command.

ip inspect alert-off

no ip inspect alert-off

Syntax Description

This command has no arguments or keywords.

Defaults

Alert messages are displayed.

Command Modes

Global configuration

Command History

Release	Modification
12.0(5)T	This command was introduced.

Usage Guidelines

Use the **ip inspect alert-off** command to disable alert messages.

Examples

The following example turns off CBAC alert messages:

```
no ip inspect alert-off
```

Debug Commands

This section documents the modified **debug** command related to the CBAC feature.

- **debug ip inspect**

debug ip inspect

To display messages about CBAC event, use the **debug ip inspect EXEC** command. The **no** form of this command disables debugging output.

```
debug ip inspect {function-trace | object-creation | object-deletion | events | timers |
  protocol | detailed}
```

```
no debug ip inspect detailed
```

Syntax Description

function-trace	Displays messages about software functions called by CBAC.
object-creation	Displays messages about software objects being created by CBAC. Object creation corresponds to the beginning of CBAC-inspected sessions.
object-deletion	Displays messages about software objects being deleted by CBAC. Object deletion corresponds to the closing of CBAC-inspected sessions.
events	Displays messages about CBAC software events, including information about CBAC packet processing.
timers	Displays messages about CBAC timer events such as when a CBAC idle timeout is reached.
<i>protocol</i>	Displays messages about CBAC-inspected protocol events, including details about the protocol's packets. Refer to Table 5 for a list of <i>protocol</i> keywords.
detailed	Use this form of the command in conjunction with other CBAC debugging commands. This causes detailed information to be displayed for all the other enabled CBAC debugging.

Table 5 Protocol Keywords for the debug ip inspect Command

Application Protocol	<i>protocol</i> keyword
Transport Layer Protocols	
TCP	tcp
UDP	udp
Application-Layer Protocols	
CU-SeeMe	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the FTP tokens parsed)	ftp-tokens
H.323	h323
HTTP	http
Microsoft NetShow	netshow
UNIX r-commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio

Table 5 Protocol Keywords for the debug ip inspect Command (continued)

Application Protocol	<i>protocol keyword</i>
RPC	rpc
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Command History

Release	Modification
11.2P	This command was introduced.
12.0(5)T	Introduced NetShow support.

Examples

The following is sample output from the **debug ip inspect function-trace** command:

```

*Mar 2 01:16:16: CBAC FUNC: insp_inspection
*Mar 2 01:16:16: CBAC FUNC: insp_pre_process_sync
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar 2 01:16:16: CBAC FUNC: insp_find_pregen_session
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_get_irc_of_idb
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_create_sis
*Mar 2 01:16:16: CBAC FUNC: insp_inc_halfopen_sis
*Mar 2 01:16:16: CBAC FUNC: insp_link_session_to_hash_table
*Mar 2 01:16:16: CBAC FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC FUNC: insp_listen_state
*Mar 2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar 2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar 2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar 2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar 2 01:16:16: CBAC FUNC: insp_process_syn_packet
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar 2 01:16:16: CBAC FUNC: insp_create_tcp_host_entry
*Mar 2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar 2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar 2 01:16:16: CBAC FUNC: insp_dec_halfopen_sis
*Mar 2 01:16:16: CBAC FUNC: insp_remove_sis_from_host_entry
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41

```

This output shows the functions called by CBAC as a session is inspected. Entries with an asterisk (*) after the word “CBAC” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect object-creation** and **debug ip inspect object-deletion** command:

```
*Mar 2 01:18:30: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar 2 01:18:30: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:30: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar 2 01:18:30: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item
25A3634
*Mar 2 01:18:31: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1
```

The following is sample output from the **debug ip inspect object-creation**, **debug ip inspect object-deletion**, and **debug ip inspect events** commands:

```
*Mar 2 01:18:51: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar 2 01:18:51: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:51: CBAC Src 10.1.0.1 Port [1:65535]
*Mar 2 01:18:51: CBAC Dst 10.0.0.1 Port [46406:46406]
*Mar 2 01:18:51: CBAC Pre-gen sis 25A3574 created: 10.1.0.1[1:65535]
30.0.0.1[46406:46406]
*Mar 2 01:18:51: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar 2 01:18:51: CBAC sis 25C1CC4 initiator_addr (10.1.0.1:20) responder_addr
(30.0.0.1:46406) initiator_alt_addr (40.0.0.1:20) responder_alt_addr (10.0.0.1:46406)
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar 2 01:18:51: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item
25A3634
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1
```

The following is sample output from the **debug ip inspect timers** command:

```
*Mar 2 01:19:15: CBAC Timer Init Leaf: Pre-gen sis 25A3574
*Mar 2 01:19:15: CBAC Timer Start: Pre-gen sis 25A3574 Timer: 25A35D8 Time: 30000
milisecs
*Mar 2 01:19:15: CBAC Timer Init Leaf: sis 25C1CC4
*Mar 2 01:19:15: CBAC Timer Stop: Pre-gen sis 25A3574 Timer: 25A35D8
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 30000 milisecs
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 3600000 milisecs
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 5000 milisecs
*Mar 2 01:19:15: CBAC Timer Stop: sis 25C1CC4 Timer: 25C1D5C
```

The following is sample output from the **debug ip inspect tcp** command:

```
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:43: CBAC sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 2544374 TCP P ack 4200176247 seq 4223720032(30)
(10.0.0.1:46409) <= (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 25412F8 TCP P ack 4223720062 seq 4200176247(15)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC sis 25C1CC4 pak 2544734 TCP S seq 4226992037(0) (10.1.0.1:20) =>
(10.0.0.1:46411)
*Mar 2 01:20:43: CBAC* sis 25C1CC4 pak 2541E38 TCP S ack 4226992038 seq 4203405054(0)
(10.1.0.1:20) <= (10.0.0.1:46411)
```

This sample shows TCP packets being processed, and lists the corresponding acknowledge (ACK) packet numbers and sequence (SEQ) numbers. The number of data bytes in the TCP packet is shown in parentheses—for example, (22). For each packet shown, the addresses and port numbers are shown separated by a colon. For example, (10.1.0.1:21) indicates an IP address of 10.1.0.1 and a TCP port number of 21.

Entries with an asterisk (*) after the word “CBAC” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect tcp** and **debug ip inspect detailed** commands:

```
*Mar 2 01:20:58: CBAC* Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar 2 01:20:58: P ack 4223720160 seq 4200176262(22)
*Mar 2 01:20:58: CBAC* Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar 2 01:20:58: CBAC* sis 25A3604 SIS_OPEN
*Mar 2 01:20:58: CBAC* Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1),
len 76,proto=6
*Mar 2 01:20:58: CBAC sis 25A3604 Saving State: SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt
4200176262 r_sndnxt 4223720160 r_rcvwnd 8760
*Mar 2 01:20:58: CBAC* sis 25A3604 pak 2541E38 TCP P ack 4223720160 seq 4200176262(22)
(30.0.0.1:46409) => (40.0.0.1:21)
*Mar 2 01:20:58: CBAC* sis 25A3604 pak 2541E38 SIS_OPEN/ESTAB TCP seq 4200176262(22)
Flags: ACK 4223720160 PSH
*Mar 2 01:20:58: CBAC* sis 25A3604 pak 2541E38 --> SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223720160 i_sndnxt 4200176284 i_rcvwnd 8760 risn 4223719771 r_rcvnxt
4200176262 r_sndnxt 4223720160 r_rcvwnd 8760
*Mar 2 01:20:58: CBAC* sis 25A3604 L4 inspect result: PASS packet 2541E38
(30.0.0.1:46409) (40.0.0.1:21) bytes 22 ftp
*Mar 2 01:20:58: CBAC sis 25A3604 Restoring State: SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223
720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262 r_sndnxt
4223720160 r_rcvwnd 8760
*Mar 2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:58: CBAC* Bump up: inspection requires the packet in the process
path(30.0.0.1) (40.0.0.1)
*Mar 2 01:20:58: CBAC Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar 2 01:20:58: P ack 4223720160 seq 4200176262(22)
*Mar 2 01:20:58: CBAC Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar 2 01:20:58: CBAC sis 25A3604 SIS_OPEN
*Mar 2 01:20:58: CBAC Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1),
len 76, proto=6
```