



The Cisco IOS Firewall Feature Set and Context-Based Access Control

This document describes the Cisco IOS Firewall feature set, and describes how to configure context-based access control, one of the Cisco IOS Firewall feature set features.

In This Document

This document includes the following sections:

- The Cisco IOS Firewall Feature Set
- Context-Based Access Control:
 - About Context-Based Access Control
 - Configure Context-Based Access Control
 - Interpret Syslog and Console Messages Generated by Context-Based Access Control
 - Turn Off Context-Based Access Control
 - CBAC Configuration Example
 - Context-Based Access Control Command Reference
 - Context-Based Access Control Debug Command

The Cisco IOS Firewall Feature Set

The Cisco IOS Firewall feature set combines existing Cisco IOS firewall technology and the new context-based access control (CBAC) feature. When you configure the Cisco IOS Firewall feature set on your Cisco router, you turn your router into an effective, robust firewall.

The Cisco IOS Firewall feature set is designed to prevent unauthorized, external individuals from gaining access to your internal network, and to block attacks on your network, while at the same time allowing authorized users to access network resources.

You can use the Cisco IOS Firewall feature set to configure your Cisco IOS router as:

- An Internet firewall or part of an Internet firewall
- A firewall between groups in your internal network
- A firewall providing secure connections to or from branch offices
- A firewall between your company's network and your company's partners' networks

The Cisco IOS Firewall feature set provides the following benefits:

- Protects internal networks from intrusion
- Monitors traffic through network perimeters
- Enables network commerce via the World Wide Web

Configuring the Cisco IOS Firewall Feature Set

To create a firewall customized to fit your organization's security policy, you should determine which features of the Cisco IOS Firewall feature set are appropriate, and configure those features. At a minimum, you must configure basic traffic filtering to provide a basic firewall. The Cisco IOS Firewall feature set includes the following features (described in the sections that follow):

- Basic and Advanced Traffic Filtering
- Security Server Support
- Network Address Translation
- Cisco Encryption Technology
- IPSec Network Security
- Neighbor Router Authentication
- Event Logging

As well as configuring these features, you should follow the guidelines listed in the section "Other Guidelines for Configuring Your Firewall." This section outlines important security practices to protect your firewall and network.

Note Refer to the Cisco IOS Release 11.3 *Security Configuration Guide* and *Security Command Reference* publications to find the complete configuration and command information for all the firewall elements described in this section, except as noted. (In particular, context-based access control—not available in Release 11.3—is described later in this document.)

Basic and Advanced Traffic Filtering

To configure traffic filtering, configure one or more of the following features:

- **Basic Traffic Filtering: Standard Access Lists and Static Extended Access Lists**

Standard and static extended access lists provide basic traffic filtering capabilities. You configure criteria that describe which packets should be forwarded, and which packets should be dropped at an interface, based on each packet's network layer information. For example, you can block all UDP packets from a specific source IP address or address range. Some extended access lists can also examine transport layer information to determine whether to block or forward packets.

- **Advanced Traffic Filtering: Lock-and-Key (Dynamic Access Lists)**

Lock-and-Key provides traffic filtering with the ability to allow temporary access through the firewall for certain individuals. These individuals must first be authenticated (by a username/password mechanism) before the firewall allows their traffic through the firewall. Afterwards, the firewall closes the temporary opening. This provides tighter control over traffic at the firewall than with standard or static extended access lists.

- **Advanced Traffic Filtering: Context-Based Access Control**

Context-based access control (CBAC) examines not only network layer and transport layer information, but also examines the application-layer protocol information (such as FTP information) to learn about the state of TCP and UDP connections. CBAC maintains connection state information for individual connections. This state information is used to make intelligent decisions about whether packets should be permitted or denied, and dynamically creates and deletes temporary openings in the firewall.

CBAC is described in greater detail later in this document.

Security Server Support

The Cisco IOS Firewall feature set can be configured as a client of the following supported security servers:

- TACACS, TACACS+, and Extended TACACS
- RADIUS
- Kerberos

You can use any of these security servers to store a database of user profiles. To gain access into your firewall or to gain access through the firewall into another network, users must enter authentication information (such as a username and password), which is matched against the information on the security server. When users pass authentication, they are granted access according to their specified privileges.

Network Address Translation

You can use Network Address Translation (NAT) to hide internal IP network addresses from the world outside the firewall.

NAT was designed to provide IP address conservation and for internal IP networks that have unregistered (not globally unique) IP addresses: NAT translates these unregistered IP addresses into legal addresses at the firewall. NAT can also be configured to advertise only one address for the entire internal network to the outside world. This provides security by effectively hiding the entire internal network from the world.

NAT gives you limited spoof protection because internal addresses are hidden. Additionally, NAT removes all your internal services from the external name space.

Note NAT does not work with the application-layer protocols RPC, VDOLive, or SQL*Net “Redirected.” (NAT does work with SQL*Net “Bequeathed.”) Do not configure NAT with networks that will carry traffic for these incompatible protocols.

To configure NAT, refer to the “Configuring IP Addressing” chapter in the Cisco IOS Release 11.3 *Network Protocols Configuration Guide, Part 1*.

Cisco Encryption Technology

Cisco encryption technology selectively encrypts IP packets that are transmitted across unprotected networks such as the Internet. You specify which traffic is considered sensitive and should be encrypted. This encryption prevents sensitive IP packets from being intercepted and read or tampered with.

IPSec Network Security

IPSec is a framework of open standards developed by the Internet Engineering Task Force (IETF) that provides security for transmission of sensitive information over unprotected networks such as the Internet. IPSec acts at the network layer, protecting and authenticating IP packets between participating IPSec devices (“peers”) such as Cisco routers.

IPSec services are similar to those provided by Cisco Encryption Technology, a proprietary security solution introduced in Cisco IOS Software Release 11.2. (The IPSec standard was not yet available at Release 11.2.) However, IPSec provides a more robust security solution, and is standards-based.

For more information related to IPSec Network Security, refer to the feature module. You can access the *IPSec Network Security* feature module on either Cisco Connection Online (CCO) or the Documentation CD-ROM.

- On CCO, the path is Software and Support, Documentation, Cisco Documentation, Cisco Product Documentation, Cisco IOS Software Configuration, Cisco IOS Release 11.3, Cisco IOS 11.3T New Features, 11.3(3)T New Features.
- On the Documentation CD, the path is Cisco Product Documentation, Cisco IOS Software Configuration, Cisco IOS Release 11.3, Cisco IOS 11.3T New Features, 11.3(3)T New Features.

Neighbor Router Authentication

Neighbor router authentication requires the firewall to authenticate all neighbor routers before accepting any route updates from that neighbor. This ensures that the firewall receives legitimate route updates from a trusted source.

Event Logging

Event logging automatically logs output from system error messages and other events to the console terminal. You can also redirect these messages to other destinations such as virtual terminals, internal buffers, or syslog servers. You can also specify the severity of the event to be logged, and you can configure the logged output to be timestamped. The logged output can be used to assist real-time debugging and management, and to track potential security breaches or other nonstandard activities throughout a network.

To configure event logging, refer to the “Troubleshooting the Router” chapter in the “System Management” part of the Cisco IOS Release 11.3 *Configuration Fundamentals Configuration Guide*.

Other Guidelines for Configuring Your Firewall

This section includes guidelines for configuring your firewall.

- 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.
- Don't enable any local service (such as SNMP or NTP) that you don't use. Cisco Discovery Protocol (CDP) and Network Time Protocol (NTP) are on by default, and you should turn these off if you don't 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.

- 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 don't already have NAT configured to prevent internal addresses from being revealed).
- Keep the firewall in a secured (locked) room.

About Context-Based Access Control

This section describes:

- What CBAC Does (Overview)
- What CBAC Does Not Do
- Platforms
- How CBAC Works
- When and Where to Configure CBAC
- The CBAC Process
- Supported Protocols
- Restrictions
- Memory and Performance Impact

What CBAC Does (Overview)

Context-based access control (CBAC) intelligently filters TCP and UDP packets based on application-layer protocol session information and can be used for intranets, extranets and internets. 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. (In other words, CBAC can inspect traffic for sessions that originate from the external network.) However, while this example discusses inspecting traffic for sessions that originate from the external network, CBAC can inspect traffic for sessions that originate from either side of the firewall.

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.

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).

CBAC also provides the following benefits:

- Java blocking
- Denial-of-Service prevention and detection
- Real-time alerts and audit trails

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 don't 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. CBAC only detects and protects against attacks that travel through the firewall.

CBAC protects against certain attacks but 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.

Platforms

The CBAC feature is supported on the following platforms:

- Cisco 1600 series
- Cisco 2500 series

How CBAC Works

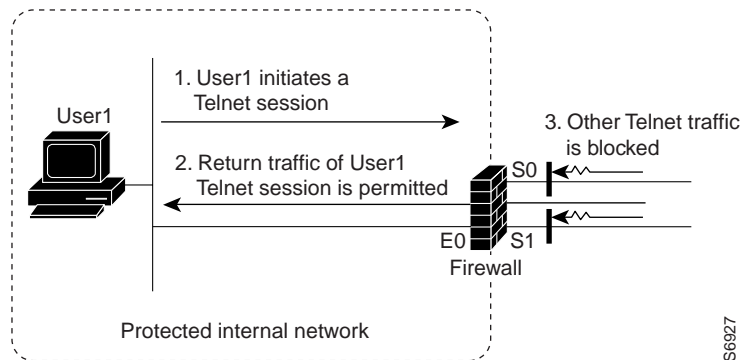
You should understand the material in this section before you configure CBAC. If you don't understand how CBAC works, you might inadvertently introduce security risks by configuring CBAC inappropriately.

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



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.

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 doesn't 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 to Permit Return Traffic and Additional Data Connections

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.

In rare cases, you might want to configure CBAC in two directions at one or more interface, which is a more complex solution. CBAC is usually only configured in two directions when the networks on both sides of the firewall should be protected, such as with extranet or intranet configurations. 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 other 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 Telnet 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.

(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.)

- 4 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.
- 5 The outbound packet is forwarded out the interface.
- 6 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.
- 7 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.
- 8 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.
- 9 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)
- Java
- UNIX R-commands (such as r-login, r-exec, and r-sh)
- RealAudio
- RPC (Sun RPC, not DCE RPC or Microsoft RPC)
- SMTP
- SQL*Net
- StreamWorks
- TFTP
- VDOLive

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

Restrictions

CBAC is available only for IP protocol traffic. Only TCP and UDP packets are inspected. (Other IP traffic, such as ICMP, cannot be filtered with CBAC and should be filtered with basic access lists instead.)

You can use CBAC together with all the other firewall features mentioned previously in the section “The Cisco IOS Firewall Feature Set.”

CBAC works with fast switching and process switching.

If you reconfigure your access lists when you configure CBAC, be aware that if your access lists block TFTP traffic into an interface, you won’t 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.

FTP Traffic and CBAC

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 won't open a data channel if the FTP client-server authentication fails.

Cisco Encryption Technology 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 target router is an endpoint 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 endpoint, 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 UDP and TCP packets.

Memory and Performance Impact

Using CBAC uses less than approximately 600 bytes of memory per connection. Because of the memory usage, you should use CBAC only when you need to. There is also a slight amount of additional processing that occurs whenever packets are inspected.

Sometimes CBAC must evaluate long access lists, which might have presented a negative impact to performance. However, this impact is avoided, because CBAC evaluates access lists using an accelerated method (CBAC hashes access lists and evaluates the hash).

Configure Context-Based Access Control

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. You should be sure you understand what CBAC does before you configure CBAC.

To configure CBAC, you must complete the tasks described in these sections:

- Pick an Interface: Internal or External
- Configure IP Access Lists at the Interface
- Configure Global Timeouts and Thresholds
- Define an Inspection Rule
- Apply the Inspection Rule to an Interface

You can also perform the tasks described in the following sections:

- Display Configuration, Status, and Statistics for Context-Based Access Control
- Debug Context-Based Access Control

Pick an Interface: Internal or External

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

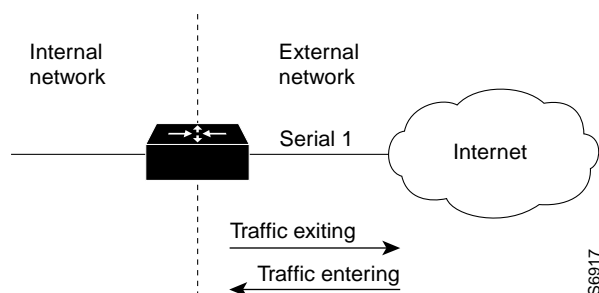
“Internal” refers to the side where sessions must originate for their traffic to be permitted through the firewall. “External” refers to the side where sessions cannot originate (sessions originating from the external side will be blocked).

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 is rarely configured in two directions, and usually only when the firewall is between two networks that need protection from each other, such as with two partners’ networks connected by the firewall.)

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.

The first topology is shown in Figure 2. 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.

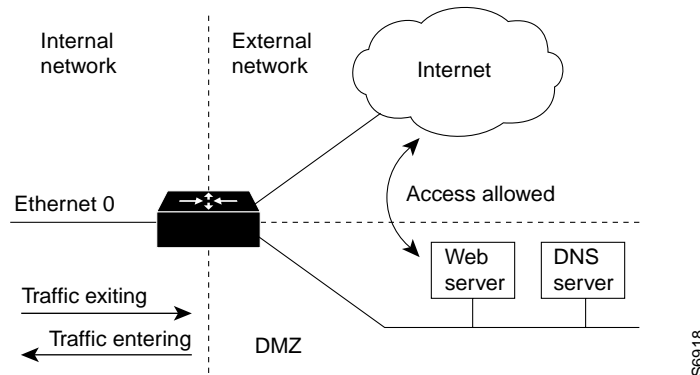
Figure 2 Simple Topology—CBAC Configured at the External Interface



S6617

The second topology is shown in Figure 3. 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.

Configure 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 two general rules when evaluating your IP access lists at the firewall:

- Permit CBAC traffic leaving 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.)

Tip: 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 they stop packets before they get a chance to affect the router itself.

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 11.3 *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 11.3 *Network Protocols Configuration Guide, Part 1*.

Configure 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 that 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 the table below).

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

To change a global timeout or threshold listed in the left column, use the global configuration command in the middle column:

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)

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 unestablished 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 unestablished 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 block-time 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 later in the section, "Context-Based Access Control 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 return any threshold or timeout to the default value, use the **no** form of the command in the preceding table.

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 rate is calculated as an exponentially-decayed rate.)

Define 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.

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

- Configure Application-Layer Protocol Inspection
- Configure Generic TCP and UDP Inspection

Configure Application-Layer Protocol Inspection

Note 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 later in the section “Configure Generic TCP and UDP Inspection.” This requirement exists because NetMeeting 2.0 uses an additional TCP channel not defined in the H.323 specification.

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

Task	Command
<p>Configure CBAC inspection for an application-layer protocol (except for RPC and Java). Use one of the protocol keywords defined in Table 1, following.</p> <p>Repeat this command for each desired protocol. Use the same <i>inspection-name</i> to create a single inspection rule.</p>	<pre>ip inspect name <i>inspection-name</i> <i>protocol</i> [timeout <i>seconds</i>]</pre>
<p>Enable CBAC inspection for the RPC application-layer protocol.</p> <p>You can specify multiple RPC program numbers by repeating this command for each program number.</p> <p>Use the same <i>inspection-name</i> to create a single inspection rule.</p>	<pre>ip inspect name <i>inspection-name</i> rpc program-number <i>number</i> [wait-time <i>minutes</i>] [timeout <i>seconds</i>]</pre>

Refer to the description of the **ip inspect name (global configuration)** command in the “Context-Based Access Control Command Reference” section later in this document for complete information about how the command works with each application-layer protocol.

To enable CBAC inspection for Java, see the following section, “Configure Java Inspection.”

Table 1 Application Protocol Keywords

Application Protocol	<i>protocol</i> Keyword
CU-See-Me	cuseeme
FTP	ftp
H.323	h323
UNIX R commands (r-login, r-exec, r-sh)	rcmd
RealAudio	realaudio
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Configure Java Inspection

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 agreeable solution, you can use context-based access control (CBAC) 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.

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, perform the following global configuration tasks:

Task	Command
Create a standard access list that permits traffic only from friendly sites, and denies traffic from hostile sites.	ip access-list standard <i>name</i> permit ... deny ... (Use permit and deny statements as appropriate.)
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.	or access-list <i>access-list-number</i> { deny permit } <i>source</i> [<i>source-wildcard</i>]

Task	Command
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. Use the same <i>inspection-name</i> as when you specified other protocols, to create a single inspection rule.	ip inspect name <i>inspection-name</i> http [java-list <i>access-list</i>] [timeout <i>seconds</i>]



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.

Configure 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, perform one or both of the following global configuration tasks:

Task	Command
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.	ip inspect name <i>inspection-name</i> tcp [timeout <i>seconds</i>]
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.	ip inspect name <i>inspection-name</i> udp [timeout <i>seconds</i>]

Apply the Inspection Rule to an Interface

After you define an inspection rule, you apply this 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.

If you are configuring CBAC on an external interface, apply the rule to outbound traffic.

If you are configuring CBAC on an internal interface, apply the rule to inbound traffic.

To apply an inspection rule to an interface, perform the following interface configuration task:

Task	Command
Apply an inspection rule to an interface.	ip inspect <i>inspection-name</i> { in out }

Display Configuration, Status, and Statistics for Context-Based Access Control

You can view certain context-based access control (CBAC) information by performing one or more of the following EXEC commands:

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

Debug Context-Based Access Control

To assist CBAC debugging, you can turn on audit trail messages which will be displayed on the console after each CBAC session closes.

To turn on audit trail messages, perform the following global configuration task:

Task	Command
Turn on CBAC audit trail messages.	ip inspect audit trail

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

Generic Debug Commands

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

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

Transport Level Debug Commands

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

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

Application Protocol Debug Commands

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

Task	Command
Display messages about CBAC-inspected protocol events, including details about the protocol's packets.	debug ip inspect <i>protocol</i>
Refer to Table 2 to determine the protocol keyword.	

Table 2 Application Protocol Keywords for the debug ip inspect Command

Application Protocol	<i>protocol</i> keyword
CU-See-Me	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the ftp tokens parsed)	ftp-tokens
H.323	h323
Java applets	http
UNIX R commands (r-login, r-exec, r-sh)	rcmd
RealAudio	realaudio
RPC	rpc
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Interpret Syslog and Console Messages Generated by Context-Based Access Control

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. While they are generally referred to as error messages, not all error messages indicate problems with your system.

The following types of error messages can be generated by CBAC:

- Denial-of-Service Attack Detection Error Messages
- SMTP Attack Detection Error Message
- Java Blocking Error Message
- FTP Error Messages
- Audit Trail Error Message

Denial-of-Service Attack Detection Error Messages

CBAC detects and blocks denial-of-service attacks and notifies you when denial-of-service attacks occur. The following error messages may indicate that denial-of-service attacks have occurred:

Error Message

```
%FW-4-ALERT_ON:[chars], count ([dec]/[dec]) current 1-min rate: [dec]
```

Explanation Either the **max-incomplete high** threshold of half-open connections or the new connection initiation rate has been exceeded. This error message indicates that an unusually high rate of new connections is coming through the firewall, and a DOS attack may be in progress. This message is issued only when the **max-incomplete high** threshold is crossed.

Recommended Action This message is for informational purposes only, but may indicate a security problem.

The following is an example of this type of message:

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

Error Message

```
%FW-4-ALERT_OFF:[chars], count ([dec]/[dec]) current 1-min rate: [dec]
```

Explanation Either the number of half-open connections or the new connection initiation rate has gone below the **max-incomplete low** threshold. This message indicates that the rate of incoming new connections has slowed down and is issued only when the **max-incomplete low** threshold is crossed.

Recommended Action This message is for informational purposes only, but may indicate that an attack has stopped.

The following is an example of this type of message:

```
%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 following example shows two separate attacks:

```
%FW-4-ALERT_ON: getting aggressive, count (25/25) current 1-min rate: 103
%FW-4-ALERT_OFF: calming down, count (9/10)current 1-min rate: 108
%FW-4-ALERT_ON: getting aggressive, count (25/25) current 1-min rate: 99
%FW-4-ALERT_OFF: calming down, count (9/10)current 1-min rate: 99
```

The following error messages may indicate that a denial-of-service attack has occurred on a specific TCP host:

Error Message

```
%FW-4-HOST_TCP_ALERT_ON: Max tcp half-open connections ([dec]) exceeded for host [int]
```

Explanation The **max-incomplete host** limit of half-open TCP connections has been exceeded. This message indicates that a high number of half-open connections is coming to the protected server, and may indicate that a SYN flood attack is in progress and is targeted to the specified server host.

Recommended Action This message is for informational purposes only, but may indicate that a SYN flood attack was attempted. If this alert is issued frequently and identified to be mostly false alarms, then the **max-incomplete host** threshold value is probably set too low, and there is a lot of legitimate traffic coming in to that server. In this case, the **max-incomplete host** parameter should be set to a higher number to avoid false alarms.

The following is an example of this type of message:

```
%FW-4-HOST_TCP_ALERT_ON: Max tcp half-open connections (50) exceeded for host 172.21.127.242.
```

For this example, the **max-incomplete host** number is set to 50 half-open sessions using the **ip inspect tcp max-incomplete host** command.

Error Message

```
%FW-2-BLOCK_HOST: Blocking new TCP connections to host [int] for [dec] minute [chars] (half-open count [dec] exceeded)
```

Explanation This message indicates that any subsequent new TCP connection attempts to the specified host will be denied because the **max-incomplete host** threshold of half-open TCP connections is exceeded, and the blocking option is configured to block the subsequent new connections. The blocking will be removed when the configured **block-time** expires.

Recommended Action This message is for informational purposes only, but may indicate that a SYN flood attack was attempted.

The following is an example of this type of message:

```
%FW-4-BLOCK_HOST: Blocking new TCP connections to host 172.21.127.242 for 2 minutes (half-open count 50 exceeded)
```

For this example, the **block-time** timeout is set to 2 minutes (120 seconds) and the **max-incomplete host** number is set to 50 half-open sessions.

Error Message

```
%FW-4-UNBLOCK_HOST: New TCP connections to host [int] no longer blocked
```

Explanation New TCP connection attempts to the specified host are no longer blocked. This message indicates that the blocking of new TCP attempts to the specified host has been lifted.

Recommended Action This message is for informational purposes only.

The following is an example of this type of message:

```
%FW-4-UNBLOCK_HOST: New TCP connections to host 172.21.127.242 no longer blocked
```

SMTP Attack Detection Error Message

CBAC detects and blocks SMTP attacks (illegal SMTP commands) and notifies you when SMTP attacks occur. The following error message may indicate that an SMTP attack has occurred:

Error Message

```
%FW-3-SMTP_INVALID_COMMAND: Invalid SMTP command from initiator  
([int]:[dec])
```

Explanation The CBAC code detected an invalid SMTP command in the inspected SMTP connection. This message indicates that a suspicious violation was detected that may be an attack to the mail server system. The command is rejected and the connection is reset by the firewall immediately.

Recommended Action This message is for informational purposes only, but may indicate a security problem.

The following is an example of this type of message:

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

Java Blocking Error Message

CBAC detects and selectively blocks Java applets and notifies you when a Java applet has been blocked. The following error message may indicate that a Java applet has been blocked:

Error Message

```
%FW-3-HTTP_JAVA_BLOCK: JAVA applet is blocked from ([int]:[dec]) to  
([int]:[dec])
```

Explanation A Java applet was seen in the HTTP channel, and the firewall configuration indicates that the applet from this Web site should be prohibited. The message indicates that the applet is being downloaded from one of the prohibited sites and its entrance to the protected network is not allowed. The connection is reset and the transmission of the detected applet is aborted immediately.

Recommended Action This message is for informational purposes only, but may indicate a security problem.

The following is an example of this type of message:

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

FTP Error Messages

CBAC detects and prevents certain FTP attacks and notifies you when this occurs. The following error messages may appear when CBAC detects these FTP attacks:

Error Message

```
%FW-3-FTP_PRIV_PORT: Privileged port [dec] used in [chars] -- FTP client
[int] FTP server [int]
```

Explanation An FTP client attempted to use a PORT command or the FTP server attempted to use the response to a PASV command to trick the firewall into opening access to a privileged port. This message indicates that a suspicious violation was detected from the FTP client/server attempting to modify the security policy in the firewall. The command is rejected and the connection is reset by the firewall.

Recommended Action This message is for informational purposes only, but may indicate that an attempt was made to gain access to privileged ports.

The following is an example of this type of message:

```
%FW-3-FTP_PRIV_PORT: Privileged port 1000 used in PORT command -- FTP client 10.0.0.1
FTP server 10.1.0.1
```

Error Message

```
%FW-3-FTP_SESSION_NOT_AUTHENTICATED: Command issued before the session is
authenticated -- FTP client [int] FTP server [int]
```

Explanation An FTP client attempted to use the PORT command or an FTP server attempted to use the response to a PASV command to open a data channel in the firewall prior to the client's successful authentication with the server. This is a suspicious attempt by the client/server to trick the firewall into opening a hole so that outside attackers can take advantage of the firewall opening. This message indicates that a suspicious violation was detected, and the PORT or PASV command/response is rejected by the firewall. The data channel in the firewall will not be opened until the authentication is done successfully.

Recommended Action This message is for informational purposes only, but may indicate that an illegal attempt was made to modify the firewall security policy.

The following is an example of this type of message:

```
%FW-3-FTP_SESSION_NOT_AUTHENTICATED: Command issued before the session is authenticated
-- FTP client 10.0.0.1
```

Error Message

```
%FW-3-FTP_NON_MATCHING_IP_ADDR: Non-matching address [int] used in [chars]
-- FTP client [int] FTP server [int]
```

Explanation An FTP client attempted to use a PORT command or the FTP server attempted to use the response to a PASV command to trick the firewall into opening access to a third-party host that is different from the two hosts engaged in the FTP connection. This message indicates that a suspicious violation was detected while attempting to modify the security policy in the firewall. The command is rejected and the connection is reset by the firewall.

Recommended Action This message is for informational purposes only, but may indicate that an attempt was made to grant or open access to unauthorized hosts.

The following is an example of this type of message:

```
%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 Error Message

CBAC provides the following audit trail message to record details about inspected sessions. To determine which protocol was inspected use the responder's port number. The port number follows the responder's address.

Error Message

```
%FW-6-SESS_AUDIT_TRAIL: [chars] session initiator ([int]:[dec]) sent [int]
bytes -- responder ([int]:[dec]) sent [int] bytes
```

Explanation This message documents the per-session transaction log of network activities. The message is issued at the end of each inspected session and it records the source/destination addresses and ports, as well as the number of bytes transmitted by the client and server.

Recommended Action This message is for informational purposes only.

The following are examples of this type of message:

```
%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
```

Turn Off Context-Based Access Control

If you so desire, you can turn off context-based access control (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.

CBAC Configuration Example

This sample configuration file shows 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:

- Ethernet 0 connects to the internal protected network
- Serial 0 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.
!-----
!
version 11.2
!
! The following three commands should appear in almost every config
!
service password-encryption
service udp-small-servers
no service tcp-small-servers
!
hostname fred-examplecorp-fr
!
boot system flash c1600-fw1600-l
enable secret 5 <elided>
!
username fred 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
!
!
!-----
!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.)
!

```

CBAC Configuration Example

```
interface Ethernet0
  description ExampleCorp Ethernet chez fred
  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 ip route-cache
  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 ip route-cache
  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
  no ip route-cache
  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 aren't 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
! Port 22 is SSH... encrypted, RSA-authenticated remote login. Can be used to get to
! field office host from ExampleCorp headquarters.
access-list 111 permit tcp any host 172.19.139.2 eq 22
! 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 igmp any any
! These are the ICMP types actually used...
! administratively-prohibited is useful when you're trying to figure out why
! you can't 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 unreachablees at all.
access-list 111 permit icmp any 172.19.139.0 0.0.0.7 unreachable
! 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

```

CBAC Configuration Example

```
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
```

Context-Based Access Control Command Reference

This sections documents new or modified commands. All other commands used with this feature are documented in the Cisco IOS Release 11.3 command references.

- **ip inspect audit trail**
- **ip inspect dns-timeout**
- **ip inspect (interface configuration)**
- **ip inspect max-incomplete high**
- **ip inspect max-incomplete low**
- **ip inspect name (global configuration)**
- **ip inspect one-minute high**
- **ip inspect one-minute low**
- **ip inspect tcp finwait-time**
- **ip inspect tcp idle-time**
- **ip inspect tcp max-incomplete host**
- **ip inspect tcp synwait-time**
- **ip inspect udp idle-time**
- **no ip inspect**
- **show ip inspect**

ip inspect audit trail

To turn on CBAC audit trail messages which will be displayed on the console after each CBAC session closes, use the **ip inspect audit trail** global configuration command. Use the **no** form of this command to turn off CBAC audit trail messages.

ip inspect audit trail
no ip inspect audit trail

Syntax Description

This command has no arguments or keywords.

Default

Audit trail messages are not displayed.

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

Example

The following example turns on CBAC audit trail messages:

```
ip inspect audit trail
```

Afterwards, audit trail messages such as the following are displayed.

```
%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: ftp session initiator (192.168.1.13:33194) sent 336 bytes --  
responder (192.168.129.11:21) sent 325 bytes
```

These messages are examples of audit trail messages. To determine which protocol was inspected, use the responder's port number. The port number follows the responder's address.

ip inspect dns-timeout

To specify the DNS idle timeout (the length of time a DNS name lookup session will still be managed after no activity), use the **ip inspect dns-timeout** global configuration command. Use the **no** form of this command to reset the timeout to the default of 5 seconds.

```
ip inspect dns-timeout seconds  
no ip inspect dns-timeout
```

Syntax Description

seconds Specifies the length of time a DNS name lookup session will still be managed after no activity.

Default

5 seconds

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

When the software detects a valid UDP packet for a new DNS name lookup session, if context-based access control (CBAC) inspection is configured for UDP, the software establishes state information for the new DNS session.

If the software detects no packets for the DNS session for a time period defined by the DNS idle timeout, the software will not continue to manage state information for the session.

The DNS idle timeout applies to all DNS name lookup sessions inspected by CBAC.

The DNS idle timeout value always overrides the global UDP timeout. The DNS idle timeout value also always overrides any timeouts specified for specific interfaces when you define a set of inspection rules with the **ip inspect name (global configuration)** command.

Examples

The following example sets the DNS idle timeout to 30 seconds:

```
ip inspect dns-timeout 30
```

The following example sets the DNS idle timeout back to the default (5 seconds):

```
no ip inspect dns-timeout
```

ip inspect (interface configuration)

To apply a set of inspection rules to an interface, use the **ip inspect** interface configuration command. Use the **no** form of this command to remove the set of rules from the interface.

```
ip inspect inspection-name { in | out }  
no ip inspect inspection-name { in | out }
```

Syntax Description

inspection-name Identifies which set of inspection rules to apply.

in Applies the inspection rules to inbound traffic.

out Applies the inspection rules to outbound traffic.

Default

If no set of inspection rules is applied to an interface, no traffic will be inspected by CBAC.

Command Mode

Interface configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

Use this command to apply a set of inspection rules to an interface.

Typically, if the interface connects to the external network, you apply the inspection rules to outbound traffic; alternately, if the interface connects to the internal network, you apply the inspection rules to inbound traffic.

If you apply the rules to outbound traffic, then return inbound packets will be permitted if they belong to a valid connection with existing state information. This connection had to have initiated with an outbound packet.

If you apply the rules to inbound traffic, then return outbound packets will be permitted if they belong to a valid connection with existing state information. This connection must be initiated with an inbound packet.

Example

The following example applies a set of inspection rules named `outboundrules` to an external interface's outbound traffic. This causes inbound IP traffic to be permitted only if the traffic is part of an existing session, and to be denied if the traffic is not part of an existing session.

```
interface serial0  
ip inspect outboundrules out
```

Related Commands

ip inspect name (global configuration)

ip inspect max-incomplete high

To define the number of existing half-open sessions that will cause the software to start deleting half-open sessions, use the **ip inspect max-incomplete high** global configuration command. Use the **no** form of this command to reset the threshold to the default of 500 half-open sessions.

```
ip inspect max-incomplete high number  
no ip inspect max-incomplete high
```

Syntax Description

number Specifies the number of existing half-open sessions that will cause the software to start deleting half-open sessions.

Default

500 half-open sessions

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

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. For UDP, “half-open” means that the firewall has detected traffic from one direction only.

Context-based access control (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).

The global value specified for this threshold applies to all TCP and UDP connections inspected by CBAC.

Examples

The following example causes the software to start deleting half-open sessions when the number of existing half-open sessions rises above 900, and to stop deleting half-open sessions with the number of existing half-open sessions drops below 800:

```
ip inspect max-incomplete high 900  
ip inspect max-incomplete low 800
```

Related Commands

- ip inspect max-incomplete low**
- ip inspect one-minute high**
- ip inspect one-minute low**
- ip inspect tcp max-incomplete host**

ip inspect max-incomplete low

To define the number of existing half-open sessions that will cause the software to stop deleting half-open sessions, use the **ip inspect max-incomplete low** global configuration command. Use the **no** form of this command to reset the threshold to the default of 400 half-open sessions.

```
ip inspect max-incomplete low number  
no ip inspect max-incomplete low
```

Syntax Description

number Specifies the number of existing half-open sessions that will cause the software to stop deleting half-open sessions.

Default

400 half-open sessions

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

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. For UDP, “half-open” means that the firewall only has detected traffic from one direction only.

Context-based access control (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).

The global value specified for this threshold applies to all TCP and UDP connections inspected by CBAC.

Examples

The following example causes the software to start deleting half-open sessions when the number of existing half-open sessions rises above 900, and to stop deleting half-open sessions with the number of existing half-open sessions drops below 800:

```
ip inspect max-incomplete high 900  
ip inspect max-incomplete low 800
```

Related Commands

ip inspect max-incomplete high

ip inspect one-minute high

ip inspect one-minute low

ip inspect tcp max-incomplete host

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* [**timeout** *seconds*]

or

ip inspect name *inspection-name http* [**java-list** *access-list*] [**timeout** *seconds*]

(Java protocol only)

or

ip inspect name *inspection-name rpc program-number number* [**wait-time** *minutes*]
[**timeout** *seconds*] (RPC protocol only)

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

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 3.
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 UDP timeouts but will not override the global DNS timeout.
java-list <i>access-list</i>	(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 <i>number</i>	Specifies the program number to permit. This keyword is available only for the RPC protocol.
wait-time <i>minutes</i>	(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.

Table 3 Protocol Keywords

Protocol	<i>protocol</i> Keyword
Transport-Layer Protocols	
TCP	tcp
UDP	udp

Table 3 Protocol Keywords (Continued)

Protocol	<i>protocol</i> Keyword
Application-Layer Protocols	
CU-See-Me	cuseeme
FTP	ftp
Java (see the section “Java Inspection,” following)	http
H.323 (see the section “H.323 Inspection,” following)	h323
UNIX R commands (r-login, r-exec, r-sh)	rcmd
RealAudio	realaudio
RPC (see the section “RPC Inspection,” following)	rpc
SMTP (see the section “SMTP Inspection,” following)	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Default

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

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

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 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.

Application-Layer Protocol Inspection

In general, if you configure inspection for an application-layer protocol, packets for that protocol will be permitted to exit the firewall, 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 and to determine if that packet belongs to a valid existing session.

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

Java Inspection

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 agreeable solution, you can use CBAC to filter Java applets at firewall, which allows users to download only applets residing within the firewall and trusted applets from outside the firewall.

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, HTTP on a nonstandard port, etc.

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 section “Configure Generic TCP and UDP Inspection.” 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.

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 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
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
```

Related Commands

ip inspect (interface configuration)

ip inspect one-minute high

To define the rate of new unestablished sessions that will cause the software to start deleting half-open sessions, use the **ip inspect one-minute high** global configuration command. Use the **no** form of this command to reset the threshold to the default of 500 half-open sessions.

```
ip inspect one-minute high number  
no ip inspect one-minute high
```

Syntax Description

number Specifies the rate of new unestablished TCP sessions that will cause the software to start deleting half-open sessions.

Default

500 half-open sessions

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

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. For UDP, “half-open” means that the firewall only has detected traffic from one direction only.

Context-based access control (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 included in the total number and rate measurements. Measurements are made once a minute.

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 rate is calculated as an exponentially-decayed rate.)

The global value specified for this threshold applies to all TCP and UDP connections inspected by CBAC.

Examples

The following example causes the software to start deleting half-open sessions when more than 1000 session establishment attempts have been detected in the last minute, and to stop deleting half-open sessions when fewer than 950 session establishment attempts have been detected in the last minute:

```
ip inspect one-minute high 1000  
ip inspect one-minute low 950
```

Related Commands

ip inspect one-minute low

ip inspect max-incomplete high

ip inspect max-incomplete low

ip inspect tcp max-incomplete host

ip inspect one-minute low

To define the rate of new unestablished TCP sessions that will cause the software to stop deleting half-open sessions, use the **ip inspect one-minute low** global configuration command. Use the **no** form of this command to reset the threshold to the default of 400 half-open sessions.

```
ip inspect one-minute low number  
no ip inspect one-minute low
```

Syntax Description

number Specifies the rate of new unestablished TCP sessions that will cause the software to stop deleting half-open sessions.

Default

400 half-open sessions

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

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. For UDP, “half-open” means that the firewall only has detected traffic from one direction only.

Context-based access control (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 included in the total number and rate measurements. Measurements are made once a minute.

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 rate is calculated as an exponentially-decayed rate.)

The global value specified for this threshold applies to all TCP and UDP connections inspected by CBAC.

Examples

The following example causes the software to start deleting half-open sessions when more than 1000 session establishment attempts have been detected in the last minute, and to stop deleting half-open sessions when fewer than 950 session establishment attempts have been detected in the last minute:

```
ip inspect one-minute high 1000  
ip inspect one-minute low 950
```

Related Commands

ip inspect one-minute high
ip inspect max-incomplete high
ip inspect max-incomplete low
ip inspect tcp max-incomplete host

ip inspect tcp finwait-time

To define how long a TCP session will still be managed after the firewall detects a FIN-exchange, use the **ip inspect tcp finwait-time** global configuration command. Use the **no** form of this command to reset the timeout to the default of 5 seconds.

```
ip inspect tcp finwait-time seconds  
no ip inspect tcp finwait-time
```

Syntax Description

seconds Specifies how long a TCP session will be managed after the firewall detects a FIN-exchange.

Default

5 seconds

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

When the software detects a valid TCP packet that is the first in a session, if context-based access control (CBAC) inspection is configured for the packet's protocol, the software establishes state information for the new session.

Use this command to define how long TCP session state information will be maintained after the firewall detects a FIN-exchange for the session. The FIN-exchange occurs when the TCP session is ready to close.

The global value specified for this timeout applies to all TCP sessions inspected by CBAC.

The timeout set with this command is referred to as the "finwait" timeout.

Note If the -n option is used with rsh, and the commands being executed do not produce output before the "finwait" timeout, the session will be dropped and no further output will be seen.

Examples

The following example changes the "finwait" timeout to 10 seconds:

```
ip inspect tcp finwait-time 10
```

The following example changes the "finwait" timeout back to the default (5 seconds):

```
no ip inspect tcp finwait-time
```

ip inspect tcp idle-time

To specify the TCP idle timeout (the length of time a TCP session will still be managed after no activity), use the **ip inspect tcp idle-time** global configuration command. Use the **no** form of this command to reset the timeout to the default of 3600 seconds (1 hour).

```
ip inspect tcp idle-time seconds  
no ip inspect tcp idle-time
```

Syntax Description

seconds Specifies the length of time a TCP session will still be managed after no activity.

Default

3600 seconds (1 hour)

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

When the software detects a valid TCP packet that is the first in a session, if context-based access control (CBAC) inspection is configured for the packet's protocol, the software establishes state information for the new session.

If the software detects no packets for the session for a time period defined by the TCP idle timeout, the software will not continue to manage state information for the session.

The global value specified for this timeout applies to all TCP sessions inspected by CBAC. This global value can be overridden for specific interfaces when you define a set of inspection rules with the **ip inspect name (global configuration)** command.

Note This command does not affect any of the currently defined inspection rules that have explicitly defined timeouts. Sessions created based on these rules still inherit the explicitly defined timeout value. If you change the TCP idle timeout with this command, the new timeout will apply to any new inspection rules you define or to any existing inspection rules that do not have an explicitly defined timeout. That is, new sessions based on these rules (having no explicitly defined timeout) will inherit the global timeout value.

Examples

The following example sets the global TCP idle timeout to 1800 seconds (30 minutes):

```
ip inspect tcp idle-time 1800
```

The following example sets the global TCP idle timeout back to the default of 3600 seconds (one hour):

```
no ip inspect tcp idle-time
```

ip inspect tcp max-incomplete host

To specify threshold and blocking time values for TCP host-specific denial-of-service detection and prevention, use the **ip inspect tcp max-incomplete host** global configuration command. Use the **no** form of this command to reset the threshold and blocking time to the default values.

```
ip inspect tcp max-incomplete host number block-time seconds
no ip inspect tcp max-incomplete host
```

Syntax Description

<i>number</i>	Specifies how many half-open TCP sessions with the same host destination address can exist at a time, before the software starts deleting half-open sessions to the host. Use a number from 1 to 250.
<i>seconds</i>	Specifies how long the software will continue to delete new connection requests to the host.

Default

50 half-open sessions and 0 seconds

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

An unusually high number of half-open sessions with the same destination host address could indicate that a denial-of-service attack is being launched against the host. For TCP, “half-open” means that the session has not reached the established state.

Whenever the number of half-open sessions with the same destination host address rises above a threshold (the **max-incomplete host** number), the software will delete half-open sessions according to one of the following methods:

- If the **block-time** *seconds* timeout is 0 (the default):
The software will delete the oldest existing half-open session for the host for every new connection request to the host. This ensures that the number of half-open sessions to a given host will never exceed the threshold.
- If the **block-time** *seconds* timeout is greater than 0:
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.

The software also sends syslog messages whenever the **max-incomplete host** number is exceeded, and when blocking of connection initiations to a host starts or ends.

The global values specified for the threshold and blocking time apply to all TCP connections inspected by CBAC.

Examples

The following example changes the **max-incomplete host** number to 40 half-open sessions, and changes the **block-time** timeout to 2 minutes (120 seconds):

```
ip inspect tcp max-incomplete host 40 block-time 120
```

The following example resets the defaults (50 half-open sessions and 0 seconds):

```
no ip inspect tcp max-incomplete host
```

Related Commands

ip inspect max-incomplete high

ip inspect max-incomplete low

ip inspect one-minute high

ip inspect one-minute low

ip inspect tcp synwait-time

To define how long the software will wait for a TCP session to reach the established state before dropping the session, use the **ip inspect tcp synwait-time** global configuration command. Use the **no** form of this command to reset the timeout to the default of 30 seconds.

```
ip inspect tcp synwait-time seconds  
no ip inspect tcp synwait-time
```

Syntax Description

seconds Specifies how long the software will wait for a TCP session to reach the established state before dropping the session.

Default

30 seconds

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

Use this command to define how long software will wait for a TCP session to reach the established state before dropping the session. The session is considered to have reached the established state after the session's first SYN bit is detected.

The global value specified for this timeout applies to all TCP sessions inspected by context-based access control (CBAC).

Examples

The following example changes the "synwait" timeout to 20 seconds:

```
ip inspect tcp synwait-time 20
```

The following example changes the "synwait" timeout back to the default (30 seconds):

```
no ip inspect tcp synwait-time
```

ip inspect udp idle-time

To specify the UDP idle timeout (the length of time a UDP “session” will still be managed after no activity), use the **ip inspect udp idle-time** global configuration command. Use the **no** form of this command to reset the timeout to the default of 30 seconds.

ip inspect udp idle-time *seconds*
no ip inspect udp idle-time

Syntax Description

seconds Specifies the length of time a UDP “session” will still be managed after no activity.

Default

30 seconds

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

When the software detects a valid UDP packet, if context-based access control (CBAC) inspection is configured for the packet’s protocol, the software establishes state information for a new UDP “session.” Because UDP is 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 if the packet was detected soon after another similar UDP packet.

If the software detects no UDP packets for the UDP session for the a period of time defined by the UDP idle timeout, the software will not continue to manage state information for the session.

The global value specified for this timeout applies to all UDP sessions inspected by CBAC. This global value can be overridden for specific interfaces when you define a set of inspection rules with the **ip inspect name (global configuration)** command.

Note This command does not affect any of the currently defined inspection rules that have explicitly defined timeouts. Sessions created based on these rules still inherit the explicitly defined timeout value. If you change the UDP idle timeout with this command, the new timeout will apply to any new inspection rules you define or to any existing inspection rules that do not have an explicitly defined timeout. That is, new sessions based on these rules (having no explicitly defined timeout) will inherit the global timeout value.

Examples

The following example sets the global UDP idle timeout to 120 seconds (2 minutes):

```
ip inspect udp idle-time 120
```

The following example sets the global UDP idle timeout back to the default of 30 seconds:

```
no ip inspect udp idle-time
```

no ip inspect

To turn off context-based access control (CBAC) completely at a firewall use the **no ip inspect** global configuration command.

no ip inspect

Syntax Description

This command has no arguments or keywords.

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

If you so desire, you can turn off CBAC, with the **no ip inspect** global configuration command.

Note The **no in 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.

Example

The following example turns off CBAC at a firewall:

```
no ip inspect
```

show ip inspect

To view CBAC configuration and session information, use the **show ip inspect** privileged EXEC command.

```
show ip inspect { name inspection-name | config | interfaces | session [detail] | all }
```

Syntax Description

name *inspection-name* Shows the configured inspection rule with the name *inspection-name*.

config Shows the complete CBAC inspection configuration.

interfaces Shows interface configuration with respect to applied inspection rules and access lists.

session [**detail**] Shows existing sessions that are currently being tracked and inspected by CBAC. The **detail** keyword causes additional details about these sessions to be shown.

all Shows all CBAC configuration and all existing sessions that are currently being tracked and inspected by CBAC.

Command Mode

Privileged EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.2 P.

Sample Display

The following is sample output for **show ip inspect name myinspectionrule**, where the inspection rule “myinspectionrule” is configured:

```
Inspection Rule Configuration
  Inspection name myinspectionrule
    tcp timeout 3600
    udp timeout 30
    ftp timeout 3600
```

The output shows the protocols that should be inspected by CBAC and the corresponding idle timeouts for each protocol.

The following is sample output for **show ip inspect config**:

```
Session audit trail is disabled
one-minute (sampling period) thresholds are [400:500] connections
max-incomplete sessions thresholds are [400:500]
max-incomplete tcp connections per host is 50. Block-time 0 minute.
tcp synwait-time is 30 sec -- tcp finwait-time is 5 sec
tcp idle-time is 3600 sec -- udp idle-time is 30 sec
dns-timeout is 5 sec
Inspection Rule Configuration
Inspection name myinspectionrule
    tcp timeout 3600
    udp timeout 30
    ftp timeout 3600
```

The output shows CBAC configuration, including global timeouts and thresholds and inspection rules.

The following is sample output for **show ip inspect interfaces**:

```
Interface Configuration
Interface Ethernet0
    Inbound inspection rule is myinspectionrule
        tcp timeout 3600
        udp timeout 30
        ftp timeout 3600
    Outgoing inspection rule is not set
    Inbound access list is not set
    Outgoing access list is not set
```

The following is sample output for **show ip inspect sessions**:

```
Established Sessions
Session 25A3318 (10.0.0.1:20)=>(10.1.0.1:46068) ftp-data SIS_OPEN
Session 25A6E1C (10.1.0.1:46065)=>(10.0.0.1:21) ftp SIS_OPEN
```

The output shows the source and destination addresses and port numbers (separated by colons) and shows that the session is an FTP session.

The following is sample output for **show ip inspect sessions detail**:

```
Established Sessions
Session 25A335C (40.0.0.1:20)=>(30.0.0.1:46069) ftp-data SIS_OPEN
    Created 00:00:07, Last heard 00:00:00
    Bytes sent (initiator:responder) [0:3416064] acl created 1
    Inbound access-list 111 applied to interface Ethernet1
Session 25A6E1C (30.0.0.1:46065)=>(40.0.0.1:21) ftp SIS_OPEN
    Created 00:01:34, Last heard 00:00:07
    Bytes sent (initiator:responder) [196:616] acl created 1
    Inbound access-list 111 applied to interface Ethernet1
```

The output includes times, number of bytes sent, and which access list was applied.

The following is sample output for **show ip inspect all**:

```
Session audit trail is disabled
one-minute (sampling period) thresholds are [400:500] connections
max-incomplete sessions thresholds are [400:500]
max-incomplete tcp connections per host is 50. Block-time 0 minute.
tcp synwait-time is 30 sec -- tcp finwait-time is 5 sec
tcp idle-time is 3600 sec -- udp idle-time is 30 sec
dns-timeout is 5 sec
Inspection Rule Configuration
  Inspection name all
    tcp timeout 3600
    udp timeout 30
    ftp timeout 3600
Interface Configuration
  Interface Ethernet0
    Inbound inspection rule is all
      tcp timeout 3600
      udp timeout 30
      ftp timeout 3600
    Outgoing inspection rule is not set
    Inbound access list is not set
    Outgoing access list is not set
Established Sessions
  Session 25A6E1C (30.0.0.1:46065)=>(40.0.0.1:21) ftp SIS_OPEN
  Session 25A34A0 (40.0.0.1:20)=>(30.0.0.1:46072) ftp-data SIS_OPEN
```

Context-Based Access Control Debug Command

This section documents the new CBAC **debug** command, **debug ip inspect**.

debug ip inspect

Use the **debug ip inspect** EXEC command to display messages about CBAC events. The **no** form of this command disables debugging output.

```
[no] debug ip inspect { function-trace | object-creation | object-deletion | events | timers |
                        protocol | detail }
[no] debug ip inspect detail
```

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 4 (following) for a list of <i>protocol</i> keywords.
detail	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 4 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-See-Me	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the ftp tokens parsed)	ftp-tokens
H.323	h323
UNIX R commands (r-login, r-exec, r-sh)	rcmd

Table 4 Protocol Keywords for the debug ip inspect Command (Continued)

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

Sample Display

The following is sample output for **debug ip inspect function-trace**:

```
*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 for **debug ip inspect object-creation** and **debug ip inspect object-deletion**:

```
*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 for **debug ip inspect object-creation**, **debug ip inspect object-deletion**, and **debug ip inspect events**:

```
*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)
M
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 for **debug ip inspect timers**:

```
*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 for **debug ip inspect tcp**:

```
*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 for **debug ip inspect tcp** and **debug ip inspect detailed**:

```
*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: -----
*Mar 2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:58: -----
*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
```

