

Cisco Secure Integrated Software H.323 V2 and RTSP Protocol Inspection

This feature module describes Cisco Secure Integrated Software (Cisco Secure IS, previously known as the Cisco IOS Firewall Feature Set) enhancements providing audio, video, and multimedia application support. It includes information on the benefits of the new feature, supported platforms, related documents, and so forth.

This document includes the following sections:

- Feature Overview, page 1
- Supported Platforms, page 4
- Supported Standards, MIBs, and RFCs, page 4
- Prerequisites, page 5
- Configuration Tasks, page 5
- Monitoring and Maintaining RTSP and H.323 V2 Inspection, page 9
- Configuration Examples, page 10
- Command Reference, page 10
- Debug Commands, page 20

Feature Overview

The Cisco Secure IS H.323 V2 and RTSP inspection feature provides firewall support for multimedia applications that require delivery of data with real-time properties such as audio and video conferencing. Cisco Secure IS has been enhanced to inspect these multimedia application protocols:

- Real Time Streaming Protocol (RTSP)
- H.323 Version 2 (H.323 V2)

This section briefly describes each of these protocols and explains how Cisco Secure IS performs protocol inspection.

RTSP

RTSP is the IETF standards-based protocol (RFC 2326) for control over the delivery of data with real time properties such as audio and video streams. It is useful for large-scale broadcasts and audio or video on demand streaming, and is supported by a variety of vendors of streaming audio and video multimedia, including Cisco IP/TV, RealNetworks RealAudio G2 Player, and Apple QuickTime 4 software.

The RFC allows for RTSP to run over either UDP or TCP, though all commercial RTSP servers are TCP-based, and Cisco Secure IS currently supports only TCP-based RTSP. RTSP establishes a TCP-based control connection, or channel, between the multimedia client and server. RTSP uses this channel to control commands such as “play” and “pause” between the client and server. These requests and responses are text-based and are similar to HTTP.

RTSP does not typically deliver continuous data streams over the control channel, usually relying on a UDP-based data transport protocol such as standard Real-Time Transport Protocol (RTP) to open separate channels for data and for RTP Control Protocol (RTCP) messages. Typically, RTP and RTCP channels occur in pairs, with RTP being an even numbered port, and RTCP channel being the next consecutive port.

Cisco Secure IS support for RTSP includes the following data transport modes:

- **Standard Real-Time Transport Protocol (RTP)**

RTP is an IETF standard (RFC 1889) supporting delivery of real-time data such as audio and video. RTP uses the RTP Control Protocol (RTCP) for managing the delivery of the multimedia data stream. RTCP helps with lip synchronization, quality of service (QoS) management, and transmitting lost packet information to the multimedia server. This is the normal mode of operation for Cisco IP/TV and Apple QuickTime 4 software.
- **RealNetworks Real Data Transport (RDT)**

RDT is a proprietary protocol developed by RealNetworks for data transport. This mode uses RTSP for controlling communications and uses RDT for the data connection and retransmission of lost packets. This is the normal mode of operation for the RealServer G2 from RealNetworks.
- **Interleaved (Tunnel Mode)**

In this mode, RTSP uses the control channel to tunnel RTP or RDT traffic. This has been incorporated for ease-of-use with firewalls and other network devices.
- **Synchronized Multimedia Integration Language (SMIL)**

SMIL is a layout language that enables the creation of multimedia presentations consisting of multiple elements of music, voice, images, text, video and graphics. This involves multiple RTSP control and data streams between the player and the servers. This mode is available only using RTSP and RDT. SMIL is a Proposed Specification of the World Wide Web Consortium (W3C). The RealNetworks RealServer and RealServer G2 provide support for SMIL—Cisco IP/TV and Apple QuickTime 4 do not.

H.323 V2

H.323 is an International Telecommunications Union (ITU) recommendation that sets standards for multimedia communications including audio and video conferencing. Cisco Secure IS supports H.323 inspection, including H.323 Version 2 and H.323 Version 1. H.323 V2 provides additional options over H.323 V1, including a “fast start” option. H.323 V2 inspection is backward compatible with H.323 V1.

With H.323 V1, a TCP connection is established between the client and server, while a separate channel for media control is negotiated using the H.245 protocol (defined in the ITU standard). With H.323 V2, audio, video, and data exchange information is included in the main control channel; there is no negotiation of a separate media channel using the H.245 protocol. This fast start option minimizes the delay between the time that a user initiates a connection and the time that the user gets the data (voice, video).

How it Works

Cisco Secure IS uses Context-based Access Control (CBAC) to inspect network traffic for access through the firewall. CBAC examines not only network layer and transport layer information, but also examines the application-layer protocol information (such as RTSP information) to learn about the dynamic UDP and TCP connections it needs to open to permit packets on those new connections, which will allow the RSTP and H323 Protocols to operate properly.

CBAC maintains the 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.

Understanding the particular TCP or UDP ports used for communication between the client and the server is important for the administration and verification of multimedia traffic through the firewall. The RTSP client uses TCP port 554 or 8554 to open a multimedia connection with a server. The data channel or data control channel (using RTCP) between the client and the server is dynamically negotiated between the client and the server using any of the high UDP ports (1024 to 65536).

The H.323 V2 client uses TCP port 1720 to open a multimedia connection with the server. The data channel between the client and the server is dynamically negotiated between the client and the server using any of the high UDP ports (1024 to 65536).

CBAC uses this port information along with connection information from the client to create dynamic access control list (ACL) entries in the firewall. As TCP or UDP connections are terminated, CBAC removes these dynamic entries from the appropriate ACLs.

Compatibility

Support for the RTSP and H.323 V2 protocols is compatible with the IP Security (IPSec) features of Cisco Secure IS.

Benefits

Multimedia Application Support

RTSP and H.323 V2 inspection allows clients on a protected network to receive data associated with a multimedia session from a server on an unprotected network.

H.323 Version 2 Enhancement Support

H.323 V2 inspection provides support for protocol enhancements associated with H.323 V2, including the fast start option.

Restrictions

Cisco Secure IS H.323 V2 and RTSP protocol inspection supports the following multimedia client-server applications:

- Cisco IP/TV
- RealNetworks RealAudio G2 Player
- Apple QuickTime 4

Related Features and Technologies

This feature module describes protocol inspection enhancements for the Cisco Secure IS CBAC feature. For the latest CBAC documentation, refer to the “Related Documents” section in this document.

Related Documents

- *Cisco IOS Firewall Feature Set*, Cisco IOS Release 12.0(5)T
- RFC 2326, *Real Time Streaming Protocol (RTSP)*
- RFC 1889, *RTP: A Transport Protocol for Real-Time Applications*
- ITU-T Recommendation H.323, *Packet based multimedia communications systems*

Supported Platforms

Cisco 2600 series

Cisco 3600 series

Cisco 7100 series

Cisco 7200 series

Supported Standards, MIBs, and RFCs

Standards

ITU-T Recommendation H.323, *Packet based multimedia communications systems*

MIBs

No new or modified MIBs are supported by this feature.

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

RFCs

- RFC 2326, *Real Time Streaming Protocol (RTSP)*

- RFC 1889, *RTP: A Transport Protocol for Real-Time Applications*

Prerequisites

This document assumes that you are currently running CBAC on the firewall router and are familiar with CBAC operation, and that you are familiar with configuring access control lists (ACLs).

This document also assumes that you are familiar with the supported multimedia applications and with the configuration procedures for each application.

Configuration Tasks

See the following sections for configuration tasks for the Cisco Secure IS H.323 V2 and RTSP Inspection. Each task in the list indicates if the task is optional or required.

- Configuring H.323 V2 or RTSP Inspection (required)
- Verifying H.323 V2 or RTSP Inspection (optional)

Configuring H.323 V2 or RTSP Inspection

To configure H.323 V2 or RTSP inspection for Cisco Secure IS, follow these steps starting in global configuration mode:

Step	Command	Purpose
1	Router(config)# access-list <i>access-list-number</i> { deny permit } <i>protocol source source-wildcard destination destination-wildcard</i> [precedence <i>precedence</i>] [tos <i>tos</i>] [established] [log]	Create an extended access control list (ACL) that denies TCP and UDP protocol traffic from unprotected networks, and that permits ICMP protocol traffic as appropriate for your site. Use the any keyword to mean any source or destination address.
2	Router(config)# interface <i>type number</i>	Select the interface type and port number where you want to apply the ACL. This command enters interface configuration mode.
3	Router(config-if) ip access-group { <i>access-list-number</i> <i>name</i> } { in out }	Apply the ACL in or out at the interface depending on your site requirements
4	Router(config)# exit	Exit to global configuration mode.
5	Router(config)# ip inspect <i>name insp-name protocol</i>	Create a CBAC inspection rule for the RTSP or H.323 protocol. If you want both RTSP and H.323 V2 support in the same rule, use the same rule name.
6	Router(config)# interface <i>type number</i>	Select the interface type and port number where you want to apply the inspection rule.
7	Router(config-if)# inspect <i>name insp-name</i> { in out }	Apply the inspection rule in or out at the interface depending on your site requirements.

Verifying RTSP or H.323 V2 Inspection

To verify RTSP inspection, initiate an RTSP- or H.323-based application through the firewall. In most cases, you can tell whether Cisco Secure IS is inspecting the multimedia session because the client application is working properly. To confirm whether the Cisco Secure IS is properly inspecting traffic, use the **show ip inspect sessions** and **show ip access lists** commands. These commands display the dynamic ACL entries and the established connections for the multimedia session. The session output can vary based on the multimedia protocol and the transport mode. This section includes the following sample session output:

- RTSP with RDT
- RTSP with TCP only (Interleaved Mode)
- RTSP with SMIL
- RTSP with RTP (IP/TV)
- H.323 V2

RTSP with RDT

This example illustrates the result of the **show ip inspect sessions** command. It shows that a control channel (rtsp) and data channel (rtsp-data) are open between hosts 192.168.155.2 and 192.168.35.1:

```
router#sh ip inspect sessions
Established Sessions
  Session 616B4F1C (192.168.155.2:7548)=>(192.168.35.1:6970) rtsp-data SIS_OPEN
  Session 611E2904 (192.168.35.1:1221)=>(192.168.155.2:554) rtsp SIS_OPEN
```

The example illustrates the result of the **show ip access-list** command. It shows that two dynamic entries (permit statements) were added to ACL 100 for the multimedia session. The TCP entry creates a dynamic opening through the firewall between port 554 (RTSP protocol port) on the client and port 1221 on the server. The UDP entry creates a dynamic opening between data port 7548 on the client and data port 6970 on the server.

```
router#sh ip access-list
Extended IP access list 100
  permit udp host 192.168.155.2 eq 7548 host 192.168.35.1 eq 6970 (31 matches)
  permit tcp host 192.168.155.2 eq 554 host 192.168.35.1 eq 1221 (27 matches)
```

After closing the multimedia session, review the session output using the **show** commands to verify the firewall software has removed the dynamic entries from the configuration.

RTSP with TCP only (Interleaved Mode)

This example illustrates the result of the **show ip inspect sessions** command. It shows that only a single control channel (rtsp) is open between hosts 192.168.155.2 and 192.168.35.1. In this mode, data is tunneled through the firewall using the TCP connection to interleave RDT or RTP data.

```
router#sh ip inspect sessions
Established Sessions
  Session 611E2904 (192.168.35.1:1228)=>(192.168.155.2:554) rtsp SIS_OPEN
```

This example illustrates the result of the **show ip access-list** command. It shows that single dynamic entry (permit statement) was added to ACL 100 for the multimedia session. The TCP entry creates a dynamic opening through the firewall between port 554 (RTSP protocol port) on the client and port 1228 on the server.

```
router#sh ip access-lists
Extended IP access list 100
    permit tcp host 192.168.155.2 eq 554 host 192.168.35.1 eq 1228 (391 matches)
```

After closing the multimedia session, review the session output using the **show** commands to verify the firewall software has removed the dynamic entries from the configuration.

RTSP with SMIL

This example illustrates the result of the **show ip inspect sessions** command for RTSP using Synchronized Multimedia Integration Language (SMIL). It shows that a single control channel (rtsp) and multiple data channels (rtsp-data) are open between hosts 192.168.155.2 and 192.168.35.1. A half open session, where UDP data flows in one direction only, which is from the server to the client.

```
router#sh ip inspect sessions
Established Sessions
  Session 616CA914 (192.168.155.2:30616)=>(192.168.35.1:6974) rtsp-data SIS_OPEN
  Session 616B4E78 (192.168.35.1:1230)=>(192.168.155.2:554) rtsp SIS_OPEN
  Session 614AB61C (192.168.155.2:29704)=>(192.168.35.1:6976) rtsp-data SIS_OPEN
  Session 616CAA88 (192.168.155.2:26764)=>(192.168.35.1:6972) rtsp-data SIS_OPEN
Half-open Sessions
  Session 614AAEF0 (192.168.155.2:15520)=>(192.168.35.1:6970) rtsp-data SIS_OPENING
```

This example illustrates the result of the **show ip access-list** command. It shows that multiple dynamic entries (permit statements) were added to ACL 100 for the multimedia session. The TCP entry creates a dynamic opening through the firewall between port 554 (RTSP protocol port) on the client and port 1230 on the server. The UDP entries create dynamic openings between negotiated data ports on the client (192.168.155.2) and the server ((192.168.35.1)).

```
fw3-3640a#sh ip access-lists
Extended IP access list 100
    permit udp host 192.168.155.2 eq 29704 host 192.168.35.1 eq 6976 (182 matches)
    permit udp host 192.168.155.2 eq 30616 host 192.168.35.1 eq 6974 (268 matches)
    permit udp host 192.168.155.2 eq 26764 host 192.168.35.1 eq 6972 (4 matches)
    permit udp host 192.168.155.2 eq 15520 host 192.168.35.1 eq 6970 (12 matches)
    permit tcp host 192.168.155.2 eq 554 host 192.168.35.1 eq 1230 (41 matches)
```

After closing the multimedia session, review the session output using the **show** commands to verify the firewall software has removed the dynamic entries from the configuration.

RTSP with RTP (IP/TV)

This example illustrates the result of the **show ip inspect sessions** command for RTSP with Cisco's IP/TV application. The output shows that a single control channel (rtsp) and multiple data channels (rtsp-data) are open between hosts 192.168.2.15 and 192.168.102.23. The data channels appear as half open sessions because the UDP data flows in one direction only, which is from the server to the client.

```
router# sh ip inspect sessions
Established Sessions
  Session 611493C0 (192.168.2.15:2571)=>(192.168.102.23:8554) rtsp SIS_OPEN
Half-open Sessions
  Session 6114A22C (192.168.102.23:2428)=>(192.168.2.15:20112) rtsp-data SIS_OPENING
  Session 61149F44 (192.168.102.23:2428)=>(192.168.2.15:20113) rtsp-data SIS_OPENING
  Session 6114A0B8 (192.168.102.23:2429)=>(192.168.2.15:20115) rtsp-data SIS_OPENING
  Session 6114A3A0 (192.168.102.23:2429)=>(192.168.2.15:20114) rtsp-data SIS_OPENING
```

This example illustrates the result of the **show ip access-list** command. It shows that multiple dynamic entries (permit statements) were added to ACL 100 for the multimedia session. The TCP entry creates a dynamic opening through the firewall between port 554 (RTSP protocol port) on the client and port 1230 on the server. The UDP entries create dynamic openings between negotiated data ports on the client (192.168.2.15) and the server ((192.168.102.23)).

```
router# sh ip access-lists
Extended IP access list 100
  permit udp host 192.168.102.23 eq 2428 host 192.168.2.15 eq 20113 (11 matches)
  permit udp host 192.168.102.23 eq 2428 host 192.168.2.15 eq 20112 (256 matches)
  permit udp host 192.168.102.23 eq 2429 host 192.168.2.15 eq 20115 (11 matches)
  permit udp host 192.168.102.23 eq 2429 host 192.168.2.15 eq 20114 (4598 matches)
  permit tcp host 192.168.102.23 eq 8554 host 192.168.2.15 eq 2571 (22 matches)
```

After closing the multimedia session, review the session output using the **show** commands to verify the firewall software has removed the dynamic entries from the configuration.

H.323 V2

This example illustrates the result of the **show ip inspect sessions** command for H.323 V2. It shows a single H.323 control channel, an RTP Control Protocol channel for both audio and video data, and an RTP data channel between hosts 192.168.155.2 and 192.168.35.1.

```
Session 615E2688 (192.168.35.1:49609)=>(192.168.155.1:49609) H323-RTCP-audio SIS_OPEN
Session 615E2688 (192.168.35.1:49508)=>(192.168.155.1:49508) H323-RTP-audio SIS_OPEN
Session 615E2688 (192.168.35.1:49410)=>(192.168.155.1:49410) H323-RTP-video SIS_OPEN
Session 615E2688 (192.168.35.1:49611)=>(192.168.155.1:49611) H323-RTCP-video SIS_OPEN
Session 615E1640 (192.168.35.1:4414)=>(192.168.155.1:1720) H323 SIS_OPEN
```

This example illustrates the result of the **show ip access-list** command. It shows that multiple dynamic entries (permit statements) were added to ACL 100 for the multimedia session. The TCP entry creates a dynamic opening through the firewall between port 1720 (H.323 V2 protocol port) on the client and port 4414 on the server. The UDP entries create dynamic openings between negotiated data ports on the client (192.168.155.1) and the server (192.168.35.1).

```
sh ip access-lists
Extended IP access list 100
  permit udp host 192.168.155.1 eq 49609 host 192.168.35.1 eq 49609 (11 matches)
  permit udp host 192.168.155.1 eq 49508 host 192.168.35.1 eq 49508 (256 matches)
  permit udp host 192.168.155.1 eq 49411 host 192.168.35.1 eq 49411 (11 matches)
  permit udp host 192.168.155.1 eq 49610 host 192.168.35.1 eq 49610 (4598 matches)
  permit tcp host 192.168.155.1 eq 1720 host 192.168.35.1 eq 4414 (22 matches)
```

Troubleshooting Tips

If you are experiencing problems getting the application to work properly, check the following items:

- The client connects with the server, but is not receiving any data.

Check that you have applied the inspection rule at the proper interface. Without the inspection rule, CBAC will not create dynamic ACL entries to allow return traffic from the server.

Additionally, verify that the multimedia application software is configured properly.

- The multimedia application is working, but the client is having trouble receiving audio or video data.

Verify that the multimedia application software is properly configured.

- The multimedia application is working, but the **show** commands do not display session information.

Check that you have applied the inspection rule at the proper interface. Without the inspection rule, CBAC will not create dynamic ACL entries to allow return traffic from the server.

Verify that the ACLs are configured to block TCP and UDP traffic at the interface. Without the proper ACL configuration, traffic can pass through the firewall without being inspected.

- The multimedia application is working, but the **show** commands do not display UDP connections.

Verify that the application is not configured to support TCP only (tunneling mode). Some applications can be set to operate in tunneling mode using TCP, which might be the desired mode of operation.

- The multimedia application has been terminated but session information still appears in the **show** command display.

Session information might not be updated immediately. Wait for a moment and try the **show** command again.

Monitoring and Maintaining RTSP and H.323 V2 Inspection

Use the following commands in Privileged EXEC mode to monitor the state of dynamic TCP and UDP sessions through the firewall:

Command	Purpose
Router# show ip inspect session	Use this command to verify the state of multimedia application connections through the firewall, including the correct source and destination IP address, and the session information (TCP and UDP).
Router# show ip access lists	Use this command to verify the creation of dynamic ACLs and to verify that dynamic entries are removed after closing the multimedia application.
Router# show ip inspect name	Use this command to check the inspection rules configured on the router. An inspection rule must include the rtsp or h323 keyword before CBAC can create dynamic ACL entries through the firewall.

Configuration Examples

This section provides a configuration example for RTSP inspection. The configuration for H.323 V2 is the same, except the **rtsp** keyword must be replaced with the **h323** keyword. Configuring protocol inspection using Cisco Secure IS has four components:

- Defining an access list with the appropriate permissions.
- Applying the ACL at an interface where you want to control access.
- Defining an inspection rule that includes the protocol that you want to inspect.
- Applying the inspection rule at an interface where you want to inspect traffic.

This example looks at each of these components. For this example, RTSP protocol inspection has been configured at a router with two Ethernet interfaces. Interface Ethernet1/0 is the protected network; interface Ethernet1/1 is the unprotected network.

ACL 100 denies TCP and UDP traffic from any source or destination while permitting specific ICMP protocol traffic. The final deny statement is not required, but is included for explicitness—the final entry in any ACL is an implicit deny of all IP protocol traffic.

```
Router(config)# access-list 100 deny tcp any any
Router(config)# access-list 100 deny udp any any
Router(config)# access-list 100 permit icmp any any echo-reply
Router(config)# access-list 100 permit icmp any any time-exceeded
Router(config)# access-list 100 permit icmp any any packet-too-big
Router(config)# access-list 100 permit icmp any any traceroute
Router(config)# access-list 100 permit icmp any any unreachable
Router(config)# access-list 100 deny ip any any
```

ACL 100 is applied inbound at interface Ethernet1/1 to manage access from the unprotected network:

```
Router(config)# interface Ethernet1/1
Router(config-if)# ip access-group 100 in
!
```

An inspection rule is created for “hquers”:

```
Router(config)# ip inspect name hquers rtsp
```

The inspection rule is applied inbound at interface Ethernet1/0 to inspect traffic from users on the protected network. CBAC uses information about the connection to create dynamic entries in ACL 100.

```
Router(config)# interface Ethernet1/0
Router(config-if)# ip inspect hquers in
```

Command Reference

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

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

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

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

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

Following is an example of the **show ip inspect sessions** command in which you want the command output to begin with the first line where the expression “audio” appears:

show ip inspect sessions | begin audio

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

ip inspect name (global configuration)

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

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

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)

HTTP Inspection Syntax

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

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

RPC Inspection Syntax

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

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

Fragment Inspection Syntax

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

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

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 1.
alert {on off}	(Optional) For each inspected protocol, the generation of alert messages can be set be on or off . If no option is selected, alerts are generated based on the setting of the ip inspect alert-off command.
audit-trail {on off}	(Optional) For each inspected protocol, audit trail can be set on or off . If no option is selected, audit trail message are generated based on the setting of the ip inspect audit-trail command.
http	(Optional) Specifies the HTTP protocol for Java applet blocking.

timeout <i>seconds</i>	(Optional) To override the global TCP or UDP idle timeouts for the specified protocol, specify the number of seconds for a different idle timeout. This timeout overrides the global TCP and UPD timeouts but will not override the global DNS timeout.
java-list <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.
fragment	Specifies fragment inspection for the named rule.
max <i>number</i>	Specifies the maximum number of unassembled packets for which state information (structures) is allocated by Cisco IOS software. Unassembled packets are packets that arrive at the router interface before the initial packet for a session. The acceptable range is 50 through 10000. The default is 256 state entries. Memory is allocated for the state structures, and setting this value to a larger number may cause memory resources to be exhausted.
timeout <i>seconds</i> (fragmentation)	Configures the number of seconds that a packet state structure remains active. When the timeout value expires, the router drops the unassembled packet, freeing that structure for use by another packet. The default timeout value is one second. If this number is set to a value greater than one second, it will be automatically adjusted by the Cisco IOS software when the number of free state structures goes below certain thresholds: when the number of free states is less than 32, the timeout will be divided by 2. When the number of free states is less than 16, the timeout will be set to 1 second.

Table 1 Protocol Keywords

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

Table 1 Protocol Keywords (continued)

Protocol	<i>protocol</i> Keyword
Application-layer Protocols	
CU-SeeMe	cuseeme
FTP	ftp
Java	http
H.323 (Version 1 and Version 2)	h323
Microsoft NetShow	netshow
UNIX R commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio
RPC	rpc
RTSP	rtsp
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Defaults

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

Command Modes

Global configuration

Command History

Release	Modification
11.2P	This command was introduced.
12.0(5)T	Introduced configurable alert and audit trail, IP fragmentation checking, and NetShow protocol support.
12.0(7)T	Introduced RTSP and H.323 Version 2 protocol support.

Usage Guidelines

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

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

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

TCP and UDP Inspection

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

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

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

Application-layer Protocol Inspection

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

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

Java Inspection

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

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



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

H.323 Inspection

CBAC supports H.323 inspection, including H.323 Version 2 (H.323 V2) and H.323 Version 1. H.323 V2 includes additional options over H.323 V1, including support for the “fast start” option. H.323 V2 inspection is backward compatible with H.323 V1.

Like RTSP, H.323 establishes a control connection between the client and server, and it negotiates data channels using additional protocols (H.225 and H.245), which are defined in the ITU standard. The fast start option enables faster negotiation of the data channels, minimizing the delay between the time that a user initiates a connection and the time that the user gets the data (voice, video).

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

RPC Inspection

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

RTSP Inspection

RTSP inspection allows control over the delivery of data with real-time properties, such as audio and video streams. With Cisco Secure IS, RTSP inspection supports the following multimedia applications:

- Cisco IP/TV
- RealNetworks RealAudio G2 Player
- Apple QuickTime 4 software.

RTSP establishes the control connection, or channel, between the multimedia client and server using TCP. RTSP uses this channel to send control commands such as “play” and “record” between the client and server. RTSP does not typically deliver continuous data streams over the control channel, usually relying on a UDP-based data transport protocol such as standard Real-Time Transport Protocol (RTP) to open separate channels for data and for RTP Control Protocol (RTCP) messages.

Cisco Secure IS support for RTSP includes the following data transport modes:

- Standard Real-Time Transport Protocol (RTP)

RTP uses the RTCP for managing the delivery of the multimedia data stream. RTCP helps with lip synchronization, quality of service (QoS) management, and transmitting lost packet information to the multimedia server. This is the normal mode of operation for Cisco IP/TV and Apple QuickTime 4 software.
- RealNetworks Real Data Transport (RDT)

RDT is a proprietary protocol developed by RealNetworks for data transport. This mode uses RTSP for controlling communications and uses RDT for the data connection and retransmission of lost packets. This is the normal mode of operation for the RealServer G2 from RealNetworks.
- Interleaved Mode

In this mode, the RTSP control connection (based on TCP) can be used to tunnel RTP or RDT traffic.
- Synchronized Multimedia Integration Language (SMIL)

SMIL is a layout language that allows easy creation of multimedia presentations consisting of multiple elements of music, voice, images, text, video, and graphics. This involves multiple RTSP control and data streams between the player and the servers. This mode is available only with RTSP using RDT.

SMTP Inspection

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

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

Use of the **timeout** Keyword

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

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

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

IP Fragmentation Inspection

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

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

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

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

Examples

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

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

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

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

Related Commands

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

Debug Commands

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

- **debug ip inspect**

debug ip inspect

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

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

```
no debug ip inspect detailed
```

Syntax Description

function-trace	Displays messages about software functions called by CBAC.
object-creation	Display 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. Table 2 provides a list of <i>protocol</i> keywords.
detailed	Use this form of the command in conjunction with other CBAC debugging commands. This causes detailed information to be displayed for all the other enabled CBAC debugging.

Table 2 Protocol Keywords for the debug ip inspect Command

Application Protocol	<i>protocol</i> keyword
Transport-layer Protocols	
TCP	tcp
UDP	udp
Application-layer Protocols	
CU-SeeMe	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the FTP tokens parsed)	ftp-tokens
H.323 (version 1 and version 2)	h323
HTTP	http
Microsoft NetShow	netshow
UNIX r-commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio
RPC	rpc

Table 2 Protocol Keywords for the debug ip inspect Command (continued)

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

Command History

Release	Modification
11.2P	This command was introduced.
12.0(5)T	Introduced NetShow support.
12.0(7)T	Introduced H.323 V2 and RTSP protocol support.

Examples

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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