Cisco − Benefits and Limitations of Context−Based Access Co

Table of Contents

Benefits and Limitations of Context−Based Access Control

Contents

Introduction

The Cisco IOS® Firewall is a group of features that makes Cisco IOS Software more effective for perimeter security. The firewall features are based on inspection of data in IP packets, including data beyond the headers.

Context−based access control (CBAC) lets the router maintain a persistent state, based on information from inspected packets, and use that state information to decide which traffic should be forwarded. CBAC is the

centerpiece of the firewall feature set, and the other features in the set build on CBAC. Connection information can be gathered from CBAC and logged to a syslog server.

What is CBAC?

Cisco IOS Firewall is a packet inspection system. It is also a stateful system; it keeps information about connections that last beyond the lifetime of a single packet. CBAC is an IP−only feature. A router running CBAC recognizes Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and some higher−layer protocols, and examines packet data beyond the IP headers. CBAC tracks information learned from the packets it examines. CBAC does several important things:

- Because CBAC recognizes TCP, UDP, and some higher−layer protocols, it extracts information from the data stream about which packets must be permitted to pass the firewall in order for those protocols to function properly. For example, CBAC permits return traffic for a TCP connection once the connection is initiated, and it recognizes FTP data connections. CBAC uses this information to create exceptions to the filtering defined by access lists.

- CBAC scans for various protocol violations and suspicious commands in the protocol streams crossing the router. If a known mode of attack or a serious protocol violation is detected, the offending traffic is blocked, and an alert is logged.

- CBAC provides the basis for session−level logging and statistics. The logging feature uses information from CBAC to log connection information to a syslog server. The logs include information about communicating hosts and port numbers, and about the number of bytes transferred.

The Cisco IOS Firewall is not a proxy system; it does not act as a TCP endpoint. Incoming packets, including fragments, are passed through the router largely unmodified. CBAC augments access lists, but does not replace them. In fact, access lists are almost always needed to make CBAC useful.

CBAC blocks network traffic only when it recognizes a protocol violation or something that appears to be a security attack. This blocking is an important feature, but for configuration purposes, CBAC creates openings in the firewall. With CBAC, you use router access lists to define which sessions will be allowed to pass through the router, rather than specifying exactly which packets are to be allowed. Your access lists need only permit the first packet of an application−layer conversation. CBAC automatically creates temporary, dynamic access list entries to permit return traffic and traffic on secondary connections associated with the conversation.

Because it adds permit entries at the beginnings of access lists, CBAC actually permits more traffic than the access lists would alone. The advantage is that the traffic permitted is carefully chosen. When CBAC is used, the underlying access list can be much more restrictive than it could be without CBAC; CBAC creates only the specific exceptions that are needed.

Concepts and Terminology

Conversations

In this document, a conversation is an application−layer protocol session, such as a Telnet login, an FTP session, or an H.323 conference. The word session is not used; this is to avoid confusion between the sessions that are displayed by the **show ip inspect sessions** command and application−layer sessions.

A conversation may involve more than one pair of hosts and ports at the transport layer. For example, an FTP conversation almost always involves at least two TCP connections, one for control and one or more for data, and an H.323 conversation may involve several UDP data streams: voice, video, and white board. CBAC creates a conversation when it sees the packet that begins that conversation. With FTP, CBAC looks for a port/passive command in the byte stream to determine whether it will allow the traffic.

Channels

A channel is an opening in the firewall that permits communication between a specific port on one host and a specific port on another host. Each entry displayed by the **show ip inspect sessions** command represents a single channel. For TCP−based protocols, channels correspond directly to TCP connections; there is a channel for each TCP connection.

CBAC creates channels dynamically as required by the application protocol. For example, if an FTP client issues commands that require a data connection to be set up, CBAC opens a channel to permit that connection. This channel is part of the overall FTP conversation. The control connection also has a channel, so there are at least two channels for most FTP conversations. When the data has been transferred, CBAC automatically deletes the data channel, and when the FTP user logs out, CBAC automatically deletes the control channel.

Whenever CBAC creates a channel, it automatically modifies the access lists on the appropriate interfaces to permit the traffic for that channel. This allows the channel's traffic to pass through the router even if the access lists would not otherwise permit it. These access list modifications are temporary; they are visible in the **show access−list** display, but do not appear in the system configuration.

Inspection Sets

An inspection set is a named set of CBAC configuration parameters, defining which traffic CBAC will inspect and how it will go about doing so. All the **ip inspect name** *inspection−name* commands that use the same inspection−name value comprise a single inspection set. There can be many inspection sets defined in a single router configuration. Each inspection set may be applied to traffic incoming on one or more interfaces, or to traffic outgoing on one or more interfaces, or both. For the purposes of applying inspection sets, the direction of traffic is the direction of conversation initiation. For instance, if a Telnet connection is made through a router, packets flow in both directions. The direction in which inspection is configured is the direction of travel of the TCP synchronize/start (SYN) packet that initiates the Telnet connection, but CBAC examines the actual packets flowing in both directions.

Generic Inspection

Generic inspection lets you use CBAC for application−layer protocols whose data streams the system doesn't know how to parse. With generic TCP inspection, for example, you can permit Telnet traffic, even though

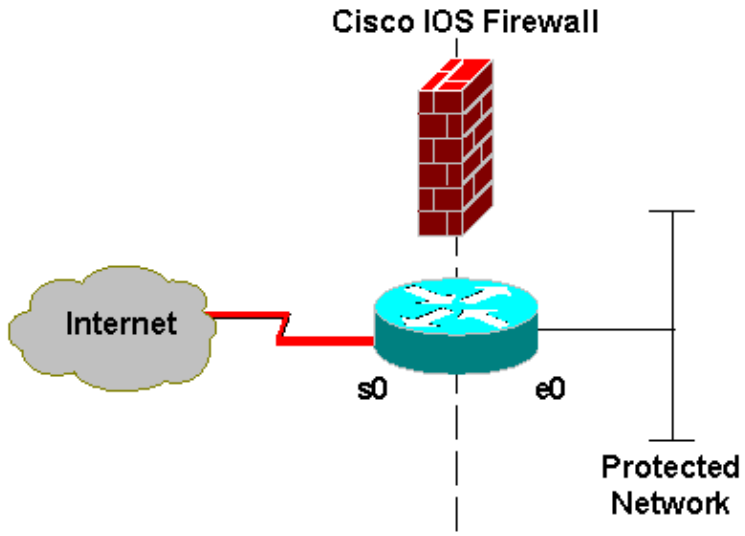
there is no inspection specific to Telnet. Generic TCP inspection works only for protocols that use a single TCP connection, initiated from the client side; it creates a single channel in response to the TCP SYN packet that opens the TCP connection, and destroys the channel in response to the TCP finish (FIN) packets that close the conversation. Generic UDP inspection works only for protocols that use a single client host/port pair and a single server host/port pair; it creates a channel in response to the first packet the client sends to the server, and leaves that channel open until there's been no activity for a specified period of time.

Benefits

Smaller Holes in Access Lists

CBAC allows you to permit less traffic than you would have to permit to get similar functionality with static access lists. For example, consider a simple firewall configuration, with a private network on Ethernet 0, and the Internet on serial 0. We want users on the private network to be able to connect to services on the Internet, but we don't want to allow any access from the Internet into the private network; the private network has absolutely no servers that the outside world needs to use. For the moment, we'll ignore real−world concerns such as anti−spoofing and return Internet Control Message Protocol (ICMP), and show some simplified partial configurations.

Diagram



Example #1− Without CBAC

Without CBAC, a simplified partial configuration might look something like this:

```
!−−− Ethernet 0 is the internal protected network.
interface ethernet 0
  ip address 10.0.0.1
!
!−−− Network traffic from the Internet comes in Serial 0.
```

```
interface serial 0
  ip address 1.1.1.1
  ip access-group 101 in
!
!--- Lets in data associated with TCP connections we've made
!--- outgoing. This opens us to some fragmentation attacks. One
!--- Linux kernel security hack makes this a potential way
!--- to make a new connection to a Linux box from outside.
access-list 101 permit tcp any any established
!
!--- Our internal Domain Name System (DNS) Server (at 10.0.0.2) may send
!--- requests to other DNS servers using port 53. This entry
!--- also allows outside machines to query our DNS.
access-list 101 permit udp any host 10.0.0.2 eq 53
!
!--- We need this access list to give our users access to
!--- UDP-based services on the outside.
access-list 101 permit udp any any gt 1024
!
!--- We need this access list to allow users to access passive ftp.
access-list 101 tcp any any gt 1023
!
access-list 101 deny ip any any
```

With this configuration, users have to be selective about which clients they use for FTP, the UNIX rsh−based commands don't work, and there are still serious security risks. Any UDP service at a port above 1024 is available. This could be improved by blocking known dangerous ports, but that is a huge administrative burden, and not very reliable. If a proxy DNS server were used, we could block all UDP ports except the ports known to be used by safe programs, but again there would be a large administrative burden, and an unsafe service could choose the same port used by a safe one.

Example #2 − Using CBAC

With CBAC, the simplified partial configuration looks something like this:

```
!--- Ethernet0 is the private internal network.
interface ethernet 0
  ip address 10.0.0.1
!
!--- Inspection is applied incoming, because connection initiation
!--- packets will be coming into this interface from the private network.
!--- Inspection always applies in the direction of conversation initiation,
!--- even though it may affect packets that are flowing in both directions.
!--- No conversations are allowed to be initiated from the Internet to the
!--- private network, so we have no inspection in that direction.
ip inspect firewall in
!
!--- Network traffic from the Internet comes in Serial0.
interface serial 0
  ip address 1.1.1.1
  ip access-group 101 in
!
!--- The access list permits nothing onto the inside network (a real
!--- list would probably allow some ICMP traffic). CBAC opens
!--- holes in response to connections initiated from the
!--- private network, so return traffic will get through.
access-list 101 deny ip any any
```

```
!
!--- Here's the list of protocols we inspect. The assumption is that
!--- we want people on the private network to be able to create any
!--- kind of outgoing conversation.
ip inspect name firewall cuseeme
ip inspect name firewall ftp
ip inspect name firewall h323
!
!--- HTTP inspection is really a no-op here, equivalent to plain TCP
!--- inspection. You could turn on Java filtering here, but it's only
!--- really recommended if you have a specific threat to block.
ip inspect name firewall http
ip inspect name firewall netshow
ip inspect name firewall rcmd
ip inspect name firewall realaudio
!
!--- Note that Simple Mail Transfer Protocol (SMTP) inspection looks
!--- for bogus commands; no auxiliary channels are needed.
ip inspect name firewall smtp
ip inspect name firewall sqlnet
ip inspect name firewall streamworks
ip inspect name firewall tftp
!
!--- Generic TCP inspection lets almost any protocol work properly, as
!--- long as it uses only a single TCP connection.
ip inspect name firewall tcp
!
!--- Generic UDP is like generic TCP: as long as there is only one client
!--- host/port and one server host/port, it works for any UDP protocol.
ip inspect name firewall udp
ip inspect name firewall vdolive
```

This configuration is long, but not complicated; most of the lines simply turn on every possible kind of inspection (except SUN Remote Procedure Call protocol [RPC]). This configuration lets users use any TCP service that uses only one connection, any UDP service that uses only one server host/port and one client host/port, and several multimedia services. Users can use any FTP client or TFTP, and they can do r−commands. At the same time, the security exposure is much less than in Example #1. TCP packets can flow from the outside only in response to traffic sent from the inside. The limitations of the established keyword are avoided.

Filtering for Protocol Abuses

CBAC recognizes some common ways of abusing protocols, and takes steps to prevent them. For example, CBAC will not allow third−party FTP hacks. When SMTP inspection is enabled, CBAC will also look in the data payload of SMTP packets for invalid commands. The valid SMTP commands are outlined in the Cisco IOS Security Command Reference. As time goes on, the number of protocol violations recognized by CBAC will increase.

Resistance Against TCP SYN Denial of Service Attacks

A TCP SYN attack occurs when an attacking source host generates TCP SYN packets with random source addresses and sends them in rapid succession to a victim host. The victim destination host sends a SYN ACK back to the random source address and adds an entry to the connection queue. Since the SYN ACK is

destined for an incorrect or nonexistent host, the acknowledgment is never completed and the entry remains in the connection queue until a timer expires. The connection queue fills up and legitimate users cannot use TCP services. However, with CBAC, TCP packets flow from the outside only in response to traffic sent from the inside. The attacking host can't get its packets through, and the attack does not succeed. In addition, by inspecting inbound on the external interface (interface serial 0 in the example above), CBAC can account for half−open connections through the firewall and begin closing those half−open connections in an agressive mode. The firewall will calm down once the number of half−open connections settles down to a user−defined value. For more information about TCP SYN DoS attacks, see Defining Strategies to Protect Against TCP SYN Denial of Service Attacks.

Synergy with Network Address Translation (NAT)

When CBAC and NAT are combined, the firewall performs CBAC inspection, and performs the following tasks:

- Hides your internal IP addresses from the outside world.
- Provides extra protection for protocols like ICMP that CBAC doesn't support.

Since NAT only allocates addresses for nodes that have actually sent traffic through the router, purely inside nodes are protected from spontaneous attack from the Internet. A node must send something to the Internet before it can be attacked.

Limitations and Caveats

Limitations of Inspection

There are two basic philosophies for a firewall that examines higher−layer traffic:

- The firewall may try to understand each supported protocol up to a certain layer, and to parse all possible commands, data formats, and options for that protocol. Once it has done this, it rejects anything it doesn't understand, treating it as a potential security violation. This approach offers high security, but lower performance, greater complexity in the firewall, a greater chance of failure when attempting to interoperate with a large number of host protocol implementations, and longer waits for users who want support for new protocol features.

- The firewall may try to understand only as much of the protocol as it absolutely needs to understand, and rejects traffic only when it sees things that are known to indicate security attacks, or protocol violations so severe that it can't parse critical information from the data stream. This approach is fast, simple, and flexible. It allows users to deploy new protocol features and unusual implementations without constant firewall upgrades. However, it risks permitting security violations that aren't known to the firewall.

Neither approach is viable in its pure form. All firewalls represent compromises between these two approaches. In most cases, the Cisco IOS Firewall favors the second approach, especially at the application layer. Because CBAC primarily guards against known modes of attack, it cannot provide protection against new or unknown modes of attack.

CBAC allows things that some administrators might consider dangerous. For example, it allows the EXPN and VRFY commands on SMTP connections. Although they are legitimate parts of SMTP, many people believe that those commands should be disabled at perimeters between administrative domains.

CBAC's relatively selective intervention in the data crossing the firewall makes it fast and flexible, but it also requires that the administrator understand what is and is not checked. Always examine the documentation before relying on the Cisco IOS Firewall to guard against any specific attack on a protocol that's being permitted to pass through the firewall. If the documentation doesn't explicitly say that something is being checked, you should assume that it's not checked, and take other steps to guard against it. You should never expose a service to the Internet, with or without CBAC, unless you believe that the software providing that service is robust and secure; don't rely on CBAC (or anything else) to completely compensate for all possible host security vulnerabilities.

Redundancy, Rerouting, and Asymmetry

All firewall states are internal to a single router, and there is no provision for redundant firewall routers. Therefore if a router running CBAC dies or is routed around, the CBAC conversations are lost.

Configurations with asymmetric routing, where only one direction of each session passes through the firewall router, do not work.

Although the Cisco IOS Firewall doesn't support router redundancy, it does support interface redundancy and load sharing. When CBAC creates a new channel, it installs the temporary access list entries on the interfaces used for the initial packet. The same access lists may be installed on backup interfaces that provide additional paths to the same destinations. It is possible to use CBAC with load sharing, as long as all the parallel interfaces are configured identically. If you configure the same access lists and inspection parameters on two interfaces that are alternate paths to the same destination, things should work more or less as expected.

Note:must use the same access lists (with the same access list numbers) on both interfaces.

Dynamic Reconfiguration

CBAC modifies access lists, which are normally considered part of the system configuration. While the changes are temporary and are not written to the configuration file, they interact with the rest of the configuration. Therefore reconfiguring access lists while CBAC is running can be dangerous. Here are a few of the failure modes:

- If an access list is deleted and recreated, CBAC temporary entries on that list are lost. Since the usual way of changing the access list on an interface is to delete and recreate the list, reconfiguring the list on an interface may lock out all the CBAC channels using that interface. If the number of the access list applied to an interface is changed, dynamic CBAC entries are not transferred to the new list. Again, CBAC conversations are lost.

- If the same access list is used on more than one interface, CBAC dynamic entries are applied to both interfaces. This can be used to support interface redundancy, so it's generally a good thing, but you shouldn't use the same access list for two interfaces unless they represent redundant paths into the same security domain. (If the two interfaces do go to the same security domain, you need to use the same access list. See the explanation above.)

- If an interface has an access list number configured with the **ip access−group** command, but the access list doesn't exist, you'll run into a peculiarity of the Cisco IOS. An empty access list usually permits everything, but a non−empty access list denies everything not explicitly permitted. When CBAC adds a temporary entry to the empty list, it becomes non−empty, thus effectively inverting its semantics. Make sure that the access lists you name in **ip access−group** commands really exist.

Timeout−Based UDP Sessions

Since UDP is fundamentally connectionless, CBAC's generic UDP inspection can't know when a client request has been satisfied. Therefore, generic UDP inspection creates a channel when it sees the first request packet from the client, and keeps that channel open until no traffic has been seen between the client and the server for a set time.

DNS UDP

CBAC does not directly inspect the Domain Name System (DNS) protocol; it uses generic UDP inspection for DNS. However, CBAC can apply a different idle timer for DNS than for other UDP transactions. This is important because a long timeout for DNS could result in a very large number of open UDP channels.

No ICMP Support

CBAC knows nothing about ICMP, either as a protocol or as a source of data that could be sent in response to a conversation using UDP or TCP. When you configure CBAC, you have to make coarse−grained decisions about which ICMP packets to allow through the firewall, just as you used to have to make coarse−grained decisions about which UDP packets to allow.

Important Information

Inspection Operates in the Direction of Conversation Initiation

The Cisco IOS Firewall has no concept of inside or outside interfaces; you can apply inspection to any traffic on any interface. The firewall uses the conversation direction. The conversation direction is the direction of the first client packet that requests establishment of that conversation. For example, if a machine on a private network connects to the Internet, the direction of the conversation is from the private network to the Internet. For CBAC to inspect this conversation (and therefore to create the access list entries allowing its traffic), the **ip inspect** command must be applied either incoming on the interface that faces the private network or outgoing on the interface that faces the Internet.

Inspection Operates Only after the Conversation is Established

CBAC won't create a conversation or open any channels unless both incoming and outgoing access lists permit the packet that initiates the conversation. This means that interface access lists can be used to control which conversations can be created. Once the conversations are created, CBAC manages them.

For example, when a connection is made from a private network to the Internet, both the incoming access list (if any) on the private network interface and the output access list on the Internet interface must permit the packet that initiates the conversation. If either list denies the packet, then the conversation is not created, and CBAC does not create any temporary access list entries.

Related Information

- **IOS Firewall Configuration Cookbook**
- **Context−Based Access Control Commands**
- **More Security Technical Tips**

Updated: Apr 02, 2002 Document ID: 10965