SECURITY

IPv6 Security

Information assurance for the next-generation Internet Protocol

Scott Hogg, CCIE® No. 5133
Eric Vyncke

ciscopress.com

IPv6 Security

Scott Hogg, CCIE No. 5133
Eric Vyncke

Cisco Press

Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

Example 4-7 *Reflexive Access List (Continued)*

```
    permit icmp any any (2 matches) sequence 20
IPv6 access list OUTREFLECT (reflexive) (per-user)
    permit tcp host 2001:DB8:22:0:20C:29FF:FEFD:F35E eq www host
       2001:DB8:11:0:6578:501F:5DF1:9C6D eq 1961 timeout 300 (6 matches) (time left
       2) sequence 1
    permit tcp host 2001:DB8:22:0:20C:29FF:FEFD:F35E eq www host
       2001:DB8:11:0:6578:501F:5DF1:9C6D eq 1964 timeout 300 (2 matches) (time left
       2) sequence 2
    permit tcp host 2001:DB8:22:0:20C:29FF:FEFD:F35E eq www host
       2001:DB8:11:0:6578:501F:5DF1:9C6D eq 1965 timeout 300 (5 matches) (time left
       2) sequence 3
    permit tcp host 2001:DB8:22:0:20C:29FF:FEFD:F35E eq www host
       2001:DB8:11:0:6578:501F:5DF1:9C6D eq 1966 timeout 300 (4 matches) (time left
       2) sequence 4
IPv6 access list INBOUND
    permit icmp any any sequence 10
    evaluate OUTREFLECT sequence 20
```

Besides the **show ipv6 access-list** command, there are several commands that you can use to manage and troubleshoot the configuration and status of the reflexive access list.

The following command clears the numbers of rule matches:

clear ipv6 access-list *access-list-name*

The following **debug** command uses the ACL to match and display packets entering the router, which is useful for troubleshooting:

debug ipv6 packet [**access-list** *access-list-name*] [**detail**]

Cisco IOS Firewall

The Cisco IOS firewall adds more substantial stateful firewalling capability to IOS compared to using traditional access lists. The IOS firewall feature set leverages Context-Based Access Control (CBAC), which adds filtering of TCP and UDP applications. CBAC inspects the traffic and creates temporary openings for the return traffic to come back through the router. This is similar to reflective ACLs but with much more granularity and awareness of connection state information as well as upper-layer protocols.

The IOS firewall has been available since IOS Release 11.3, but in IOS Release 12.3(7)T, the IOS firewall was adapted to support the IPv6 protocol. Now, the IOS firewall provides stateful protocol inspection with anomaly detection for IPv6, TCP, UDP, and ICMPv6 traffic. It also inspects upper-layer protocols such as SIP, H.323, HTTP, and FTP to check for protocol consistency and to monitor the connection state. The IOS firewall feature set has historically been offered with a combination of features. CBAC, authentication proxy, port-to-application mapping (PAM), and intrusion detection are all part of the IOS firewall feature set package.

When the IOS firewall added IPv6 support, features such as fragmented packet inspection were added, and eventually that became the Virtual Fragment Reassembly (VFR) feature. This feature inspects fragments and looks for out-of-order fragments while reassembling the fragments and inspecting the payload based on its upper-layer protocol information.

The IOS firewall also includes DoS attack monitoring and mitigation. TCP SYN flags are inspected and half-open connections (connections where the firewall sees the outbound SYN and is still waiting for the inbound SYN+ACK packet) are prevented from getting out of hand.

The IOS firewall is extension header–aware and easily parses many types of option headers. The Cisco IOS firewall can also check other IPv6 header fields such as traffic class, flow label, payload length, and hop limit, in addition to the source and destination address. IOS firewall IPv6 support also comes with the ability to inspect tunneled traffic when the tunnels terminate in the router as well as traffic that traverses both IPv6 and IPv4 environments.

One disadvantage of the IOS firewall feature set is that it requires cycles from the CPU of the network device. Because network devices do not have many extra CPU cycles to spare, this precious commodity must be used sparingly. If extensive use is made of the IOS firewall capabilities, the CPU utilization on the router will rise. The amount that is consumed depends on the complexity of the firewall configurations and the amount of traffic being filtered. The amount of logging being performed also contributes to the router's higher CPU utilization when performing the firewall function. The amount of firewall performance that can be delivered by a network device varies based on the processor it has, what other functions are being performed on that device, and the amount of traffic volume. Therefore, IOS firewall should not be used on core network devices but rather on routers at the edges of networks. If a higher level of performance is required, the Firewall Service Module (FWSM) in a 6509 chassis can provide up to 5 Gbps of filtering capability, and the ASA 5580 and ASR 1000 are good platforms to achieve up to 10 Gbps of firewall throughput.

Configuring IOS Firewall

To configure IOS firewall (CBAC), you create an inspection policy and accompanying ACLs and then apply them to the interfaces in the desired directions. Traditionally an inbound IP access list is applied to the external interface. This access list permits all packets that you want to allow to enter the network, including packets you want to be inspected by CBAC. An outbound extended IP access list is also applied to the internal interface. This access list denies any traffic to be inspected by CBAC. When CBAC is triggered with an inbound packet, CBAC creates a temporary opening in the outbound access list to permit only traffic that is part of a valid existing session.

The first step in configuring IOS firewall is to define the inspection policy. This involves using the **ipv6 inspect** command to create a name for the policy and define the various

protocols that are to be inspected. The final step is applying the inspection policy name to the interface. The syntax of these commands follows:

```
ipv6 inspect name inspection-name protocol [alert {on ¦ off}] [audit-trail {on ¦
  off}] [timeout seconds]
interface FastEthernet0/0
 ipv6 inspect inspect-name {in ¦ out}
```

The IOS firewall can perform protocol inspection on FTP, TCP, UDP, and even ICMP packets. It is clear that having stateful inspection of TCP connections is important, but the inspection of ICMP messages is vitally important in IPv6 networks. Because the protocol is heavily relied on, it could be used by attackers to spoof packets. Therefore, ICMP inspection is important to protecting an IPv6 network, both at the perimeter and in the interior. The IOS firewall also can check out routing headers with the **ipv6 inspect routing-header** command. This command uses CBAC to inspect routing header packets. If you simply want the router to drop routing header packets, you can use the **no ipv6 inspect routing-header** command.

The IOS firewall can perform DoS filtering on IPv6 packets just the same as it does for IPv4 packets. This feature helps prevent half-open connections created by SYN flood attacks from harming the infrastructure servers. To help control DoS attacks, the IOS firewall can tune its parameters that it uses to control how stateful connections are maintained. If the IOS firewall observes too many SYN packets that results in many half-open connections, it helps protect the systems by resetting the connections to accommodate new connections. There are several parameters that can be configured to control timers and to enable logging. These parameters can be tuned with the following commands based on your specific implementation and traffic patterns.

The following command allows you to set the length of time CBAC waits for a new TCP connection to reach the established state. If the SYN is received but that is not followed with a complete three-way handshake, the connection is reset. The default value is 30 seconds, but you might want to increase this to 60 seconds if you have a highly congested network. Alternatively, you might want to set it lower—to 15 seconds—to be more granular in your inspection of TCP sessions.

```
ipv6 inspect tcp synwait-time {seconds}
```

The following command sets the duration that CBAC manages a TCP session after it has been closed with the double two-way FIN exchange. The default value is 5 seconds, but you can set it to 1 second if you have a very busy network where you want to remove closed connections very quickly.

```
ipv6 inspect tcp finwait-time {seconds}
```

The following command sets the duration that CBAC manages a TCP session with no activity. The default value is for one hour (3600 seconds), but 30 minutes (1800 seconds) might be a better value to age-out idle sessions faster.

```
ipv6 inspect tcp idle-time {seconds}
```

The following command sets the duration that CBAC manages a UDP connection with no activity. The default value is 30 seconds, but you might need to increase this to 60 seconds if your network is highly congested with UDP traffic. Alternatively, you might want to set it lower—to 15 seconds—to be more granular in your inspection of UDP sessions.

```
ipv6 inspect udp idle-time {seconds}
```

You can use the following command to set the host-specific DoS prevention parameter values. This command sets the maximum number of half-open connections on a host basis. The block-time is the duration for denying any new half-open connections from a host. The default value is 50 connections, and the block-time is set to 0 minutes. With the block-time set to 0 minutes, the router keeps deleting the oldest half-open connection from a host to hold it to the limit. When the block-time is set to any value greater than 0 minutes, the router blocks any new half-open connection from a host when the maximum number of half-open connections is reached. Only when the block-time has passed can the host create any new half-open connections.

```
ipv6 inspect tcp max-incomplete host <value> [block-time <minutes>]
```

When the number of half-open connections exceeds the maximum level set in the router, the IOS firewall resets connections until the number of half-open connections falls below the low-water mark value. The router also sends syslog messages indicating that the max-incomplete number of half-open connections has been reached, that blocking has started, and also when normal operation has been restored. You can use the following commands to set the max half-open connection per host and the low-end restoration value. The default high-limit value is 500 half-open connections, and the default value of the low-limit value is set to 400. If you experience resets on normal traffic, you should set the high value to be 25 percent higher than the maximum count your network device has ever recorded.

```
ipv6 inspect max-incomplete high <value>
ipv6 inspect max-incomplete low <value>
```

The IOS firewall also keeps track of a one-minute total of the number of TCP, UDP, and ICMP connections. The default high-water mark of the number of connections allowed in one minute defaults to 500, and the low-water mark defaults to 400 connections. Following are the commands you would use to modify these parameters:

```
ipv6 inspect one-minute high <value>
ipv6 inspect one-minute low <value>
```

One technique you can use is to set these values very high and then view the statistics maintained by the IOS firewall to determine the high-water mark for your network. You should check the default settings in the IOS version you are running to see whether these are set too high or low for your particular network's traffic patterns. In some IOS versions, the default settings might be intentionally set very high to allow tuning of these parameters. This way you will not accidentally drop legitimate traffic if the default values are set too low for your environment. Then you can adjust the parameters to be effective but give your network some headroom.

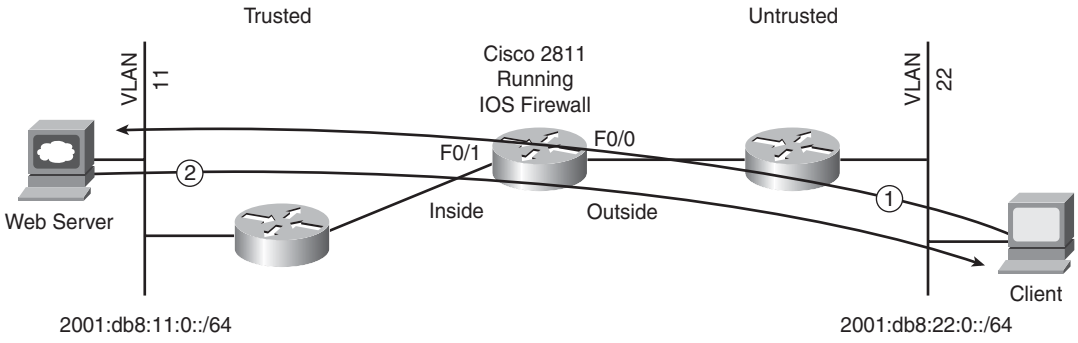
Another important IOS firewall command is **ipv6 inspect audit-trail**. This command enables the IOS firewall to log all connection attempts into the router's log. It is disabled by default to help the performance of the IOS firewall, but it can be turned on for short periods for testing. Another useful command is **no ipv6 inspect alert-off**. This command turns on the IOS firewall session alert function because it is disabled by default. If this is turned on, you will get alert log messages when illegal connections are made and dropped. Following is an example of the message that might be displayed:

```
*Dec 12 05:01:17.003: %IPV6-6-ACCESSLOGP: list FILTER-IN/40 denied tcp
2001:DB8:11:0:20C:29FF:FE50:7F0D(53950) -> 2001:DB8:22:0:20C:29FF:FEFD:F35E(80), 1
  packet
```

IOS Firewall Example

This section contains an example of how you might configure the IOS firewall for IPv6. In this scenario, there is a client computer that needs to securely access a web server with HTTP and Secure FTP (SFTP)/Secure Shell (SSH). The router must not allow computers on the same LAN as the web server being able to access the client computer. Figure 4-2 is a diagram of the router's network topology; it shows how the HTTP or SFTP connections are created.

Figure 4-2 *IOS Firewall Scenario*



First, the client computer on the untrusted network uses a web browser to connect to the internal web server. The inbound ACL that is applied to the outside untrusted interface is checked. CBAC inspects the packet as it is sent toward the destination web server. This is the point when CBAC processing occurs. Just before the first TCP SYN packet leaves the router toward the web server, the router permits the return traffic to be allowed back in on the inside interface. The TCP packet is forwarded through the inside interface to the web server. When the web server responds with a TCP SYN-ACK packet, the CBAC session table is checked and the connection is permitted. The client web session is now established, and HTTP data can be transmitted.

Example 4-8 shows the configuration example for this network. The outside interface has an ACL applied in the inbound direction to allow the HTTP and SFTP/SSH connections to the web server. The IPv6 inspect policy is named V6FW, and it is applied to the Fast Ethernet 0/0 interface on the Cisco 2811 router. This is the untrusted external interface that is closest to the client computer. The inside interface also has the IOS firewall inspection enabled so that it can see the state of the connection as it is initiated. Two ACLs are created: one for the traffic heading inbound as it enters the router's outside interface and one for the traffic that is prevented from leaving the router from the inside trusted interface. The ACL that is applied to the inside interface blocks most IPv6 packets. CBAC can watch for stateful connections that match the inbound ACL and dynamically allow the returning web traffic.

Example 4-8 *IPv6 IOS Firewall Configuration*

```
ipv6 inspect audit-trail
ipv6 inspect max-incomplete low 150
ipv6 inspect max-incomplete high 250
ipv6 inspect one-minute low 100
ipv6 inspect one-minute high 200
ipv6 inspect name V6FW tcp
ipv6 inspect name V6FW udp
ipv6 inspect name V6FW icmp
ipv6 inspect routing-header
!
interface FastEthernet0/0
 description Outside - Untrusted Side (Public)
 ipv6 traffic-filter FILTER-IN in
 ipv6 inspect V6FW in
!
interface FastEthernet0/1
 description Inside - Trusted Side (Private)
 ipv6 traffic-filter FILTER-OUT in
 ipv6 inspect V6FW in
!
ipv6 access-list FILTER-IN
 permit icmp any any
 permit tcp 2001:db8:22:0::/64 host 2001:DB8:11:0:20C:29FF:FEB8:7E50 eq www
 permit tcp 2001:db8:22:0::/64 host 2001:DB8:11:0:20C:29FF:FEB8:7E50 eq 22
 deny ipv6 any any log
!
ipv6 access-list FILTER-OUT
 permit icmp any any
 deny ipv6 any any log
```

After this is configured, the client can successfully connect to the web server using HTTP on TCP port 80, and the client can connect using SFTP or SSH over TCP port 22. You can verify the configuration with the following command:

```
show ipv6 inspect {name inspection-name ¦ config ¦ interfaces ¦ session [detail] ¦
  all}
```