Connection Settings

This chapter describes how to configure connection settings for connections that go through the ASA, or for management connections that go to the ASA.

- Connection settings, on page 1
- Configure Connection Settings, on page 2
- Monitoring Connections, on page 25
- History for Connection Settings, on page 27

Connection settings

Connection settings refer to a variety of features that manage connections through a device.

Connection setting types

Connection settings include these types:

- **Global timeouts for various protocols**—All global timeouts have default values. Change them only if you experience premature connection loss.
- **Connection timeouts per traffic class**—You can override the global timeouts for specific types of traffic using service policies. All traffic class timeouts have default values, so you do not have to set them.
- **Connection limits and TCP Intercept**—By default, there are no limits on how many connections can go through (or to) the ASA. Use service policy rules to set limits on particular traffic classes. This protects servers from denial of service attacks Particularly, you can set limits on embryonic connections (those that have not finished the TCP handshake), which protects against SYN flooding attacks. When embryonic limits are exceeded, the TCP Intercept component activates to proxy connections and throttle attacks.
- **Dead connection detection (DCD)**—Enable dead connection detection if you have persistent connections that are valid but often idle. These connections may get closed when they exceed idle timeout settings. Dead connection detection identifies idle but valid connections and keeps them alive by resetting their idle timers. Whenever idle times are exceeded, DCD probes both sides of the connection to see if both sides agree that the connection is valid.. The **show service-policy** command output includes counters to show the amount of activity from DCD. You can use the **show conn detail** command to get information about the initiator and responder and how often each has sent probes.
- **TCP sequence randomization**—Each TCP connection has two initial sequence numbers (ISN): one generated by the client and one generated by the server. By default, the ASA randomizes the ISN of the TCP SYN passing in both the inbound and outbound directions. Randomization prevents an attacker

from predicting the next ISN for a new connection and potentially hijacking the new session. However, TCP sequence randomization effectively breaks TCP SACK (Selective Acknowledgement), as the sequence numbers the client sees are different from what the server sees. You can disable randomization per traffic class if desired.

- **TCP normalization**—The TCP Normalizer protects against abnormal packets. You can configure how some types of packet abnormalities are handled by traffic class.
- **TCP state bypass**—You can bypass TCP state checking if you use asymmetrical routing in your network.
- **SCTP state bypass**—You can bypass Stream Control Transmission Protocol (SCTP) stateful inspection if you do not want SCTP protocol validation.
- **Flow offloading**—You can identify select traffic to be offloaded to a super fast path, where the flows are switched in the NIC itself. Offloading can help you improve performance for data-intensive applications such as large file transfers.
- **IPsec flow offload**—After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the field-programmable gate array (FPGA) in the device, which should improve device performance. This feature is enabled by default on platforms that support it.

Configure Connection Settings

Connection limits, timeouts, TCP Normalization, TCP sequence randomization, and decrementing time-to-live (TTL) have default values that are appropriate for most networks. You need to configure these connection settings only if you have unusual requirements, your network has specific types of configuration, or if you are experiencing unusual connection loss due to premature idle timeouts.

Other connection-related features are not enabled. You would configure these services on specific traffic classes only, and not as a general service. These features include the following: TCP Intercept, TCP State Bypass, Dead Connection Detection (DCD), SCTP state bypass, flow offload.

The following general procedure covers the gamut of possible connection setting configurations. Pick and choose which to implement based on your needs.

Before you begin

When a packet enters the device, the firewall first evaluates access control rules and NAT to determine whether a connection entry needs to be created. A connection entry is created regardless of whether an ARP entry exists for the next hop. Thus, the existence of a connection does not mean that any packet within the scope of the connection made it through the device. Connection settings such as idle timeouts ensure that unwanted connections do not linger and take up system resources.

Procedure

Step 1 Configure Global Timeouts, on page 3. These settings change the default idle timeouts for various protocols for all traffic that passes through the device. If you are having problems with connections being reset due to premature timeouts, first try changing the global timeouts.

Step 2 Protect Servers from a SYN Flood DoS Attack (TCP Intercept), on page 5. Use this procedure to configure TCP Intercept.

Step 3 Customize Abnormal TCP Packet Handling (TCP Maps, TCP Normalizer), on page 7, if you want to alter the default TCP Normalization behavior for specific traffic classes.

Step 4 Bypass TCP state checks for asymmetrical routing (TCP state bypass), on page 10, if you have this type of routing environment.

Step 5 Disable TCP Sequence Randomization, on page 12, if the default randomization is scrambling data for certain connections.

Step 6 Offload Large Flows, on page 13, if you need to improve performance in a computing intensive data center.

Step 7 Configure Connection Settings for Specific Traffic Classes (All Services), on page 22. This is a catch-all procedure for connection settings. These settings can override the global defaults for specific traffic classes using service policy rules. You also use these rules to customize TCP Normalizer, change TCP sequence randomization, decrement time-to-live on packets, and implement other optional features.

Step 8 Configure TCP Options, on page 24, if you need to force resets or change some other standard TCP behavior.

Configure Global Timeouts

You can set the global idle timeout durations for the connection and translation slots of various protocols. If the slot has not been used for the idle time specified, the resource is returned to the free pool.

Changing the global timeout sets a new default timeout, which in some cases can be overridden for particular traffic flows through service policies.

For protocols that do not have a specific configurable timeout setting, such as GRE, the idle timeout is 2 minutes.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Advanced** > **Global Timeouts**.

Step 2 Configure the timeouts by checking the boxes for timeouts you want to change and entering the new value.

All durations are displayed in the format *hh*:*mm*:*ss*, with a maximum duration of 1193:0:0 in most cases. In all cases, except for Authentication absolute and Authentication inactivity, unchecking the check boxes returns the timeout to the default value. For those two cases, clearing the check box means to reauthenticate on every new connection.

Enter 0 to disable a timeout.

- **Connection**—The idle time until a connection slot is freed. This duration must be at least 5 minutes. The default is 1 hour.
- **Half-closed**—The idle time until a TCP half-closed connection closes. A connection is considered half-closed if both the FIN and FIN-ACK have been seen. If only the FIN has been seen, the regular connection timeout applies. The minimum is 30 seconds. The default is 10 minutes.
- **UDP**—The idle time until a UDP connection closes. This duration must be at least 1 minute. The default is 2 minutes.

- **ICMP**—The idle time after which general ICMP states are closed. The default (and minimum) is 2 seconds.
- **ICMP Error**—The idle time before the ASA removes an ICMP connection after receiving an ICMP echo-reply packet, between 0:0:0 and 0:1:0 or the ICMP timeout value, whichever is lower. The default is 0 (disabled). When this timeout is disabled, and you enable ICMP inspection, then the ASA removes the ICMP connection as soon as an echo-reply is received; thus any ICMP errors that are generated for the (now closed) connection are dropped. This timeout delays the removal of ICMP connections so you can receive important ICMP errors.
- **H.323**—The idle time after which H.245 (TCP) and H.323 (UDP) media connections close. The default (and minimum) is 5 minutes. Because the same connection flag is set on both H.245 and H.323 media connections, the H.245 (TCP) connection shares the idle timeout with the H.323 (RTP and RTCP) media connection.
- **H.225**—The idle time until an H.225 signaling connection closes. The default is 1 hour. To close a connection immediately after all calls are cleared, a timeout of 1 second (0:0:1) is recommended.
- **MGCP**—The idle time after which an MGCP media connection is removed. The default is 5 minutes, but you can set it as low as 1 second.
- **MGCP PAT**—The idle time after which an MGCP PAT translation is removed. The default is 5 minutes. The minimum time is 30 seconds.
- **TCP Proxy Reassembly**—The idle timeout after which buffered packets waiting for reassembly are dropped, between 0:0:10 and 1193:0:0. The default is 1 minute (0:1:0).
- **Floating Connection**—When multiple routes exist to a network with different metrics, the system uses the one with the best metric at the time of connection creation. If a better route becomes available, then this timeout lets connections be closed so a connection can be reestablished to use the better route. The default is 0 (the connection never times out). To make it possible to use better routes, set the timeout to a value between 0:0:30 and 1193:0:0. This timer does not apply to connections through virtual tunnel interfaces (VTI). If a connection through a VTI gets stuck, you must manually clear it.
- **SCTP**—The idle time until a Stream Control Transmission Protocol (SCTP) connection closes, between 0:1:0 and 1193:0:0. The default is 2 minutes (0:2:0).
- **Stale Routes**—How long to keep a stale route before removing it from the router information base. These routes are for interior gateway protocols such as OSPF. The default is 70 seconds (00:01:10), the range is 00:00:10 to 00:01:40.
- **SUNRPC**—The idle time until a SunRPC slot is freed. This duration must be at least 1 minute. The default is 10 minutes.
- **SIP**—The idle time until a SIP signaling port connection closes. This duration must be at least 5 minutes. The default is 30 minutes.
- **SIP Media**—The idle time until a SIP media port connection closes. This duration must be at least 1 minute. The default is 2 minutes. The SIP media timer is used for SIP RTP/RTCP with SIP UDP media packets, instead of the UDP inactivity timeout.
- **SIP Provisional Media**—The timeout value for SIP provisional media connections, between 1 and 30 minutes. The default is 2 minutes.
- **SIP Invite**—The idle time after which pinholes for PROVISIONAL responses and media xlates will be closed, between 0:1:0 and 00:30:0. The default is 3 minutes (0:3:0).

- **SIP Disconnect**—The idle time after which SIP session is deleted if the 200 OK is not received for a CANCEL or a BYE message, between 0:0:1 and 0:10:0. The default is 2 minutes (0:2:0).
- **Authentication absolute**—The duration until the authentication cache times out and users have to reauthenticate a new connection. This timer is used in cut-through proxy only, which is a AAA rule. This duration must be shorter than the Translation Slot timeout. The system waits until the user starts a new connection to prompt again. Before you disable caching to force authentication on every new connection, consider the following limitations.
  - Do not set this value to 0 if passive FTP is used on the connections.
  - When Authentication Absolute is 0, HTTPS authentication may not work. If a browser initiates multiple TCP connections to load a web page after HTTPS authentication, the first connection is permitted through, but subsequent connections trigger authentication. As a result, users are continuously presented with an authentication page, even after successful authentication. To work around this, set the authentication absolute timeout to 1 second. This workaround opens a 1-second window of opportunity that might allow non-authenticated users to go through the firewall if they are coming from the same source IP address.
- **Authentication inactivity**—The idle time until the authentication cache times out and users have to reauthenticate a new connection. This duration must be shorter than the Translation Slot value. This timeout is disabled by default. This timer is used in cut-through proxy only, which is a AAA rule.
- **Translation Slot**—The idle time until a NAT translation slot is freed. This duration must be at least 1 minute. The default is 3 hours.
- **PAT Translation Slot** (8.4(3) and later, not including 8.5(1) and 8.6(1))—The idle time until a PAT translation slot is freed, between 0:0:30 and 0:5:0. The default is 30 seconds. You may want to increase the timeout if upstream routers reject new connections using a freed PAT port because the previous connection might still be open on the upstream device.
- **Connection Holddown**—How long the system should maintain a connection when the route used by the connection no longer exists or is inactive. If the route does not become active within this holddown period, the connection is freed. The purpose of the connection holddown timer is to reduce the effect of route flapping, where routes might come up and go down quickly. You can reduce the holddown timer to make route convergence happen more quickly. The default is 15 seconds, the range is 00:00:00 to 00:00:15.

Step 3 Click **Apply**.

Protect Servers from a SYN Flood DoS Attack (TCP Intercept)

A SYN-flooding denial of service (DoS) attack occurs when an attacker sends a series of SYN packets to a host. These packets usually originate from spoofed IP addresses. The constant flood of SYN packets keeps the server SYN queue full, which prevents it from servicing connection requests from legitimate users.

You can limit the number of embryonic connections to help prevent SYN flooding attacks. An embryonic connection is a connection request that has not finished the necessary handshake between source and destination.

When the embryonic connection threshold of a connection is crossed, the ASA acts as a proxy for the server and generates a SYN-ACK response to the client SYN request using the SYN cookie method, so that the connection is not added to the SYN queue of the targeted host. The SYN cookie is the initial sequence number

returned in the SYN-ACK that is constructed from MSS, time stamp, and a mathematical hash of other items to essentially create a secret. If the ASA receives an ACK back from the client with the correct sequence number and within the valid time window, it can then authenticate that the client is real and allow the connection to the server. The component that performs the proxy is called TCP Intercept.

The end-to-end process for protecting a server from a SYN flood attack involves setting connection limits, enabling TCP Intercept statistics, and then monitoring the results.

Before you begin

- Ensure that you set the embryonic connection limit lower than the TCP SYN backlog queue on the server that you want to protect. Otherwise, valid clients can no longer access the server during a SYN attack. To determine reasonable values for embryonic limits, carefully analyze the capacity of the server, the network, and server usage.
- Depending on the number of CPU cores on your ASA model, the maximum concurrent and embryonic connections can exceed the configured numbers due to the way each core manages connections. In the worst case scenario, the ASA allows up to *n*-1 extra connections and embryonic connections, where *n* is the number of cores. For example, if your model has 4 cores, if you configure 6 concurrent connections and 4 embryonic connections, you could have an additional 3 of each type. To determine the number of cores for your model, enter the **show cpu core** command.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Service Policy**.

Step 2 Click **Add** > **Add Service Policy Rule**.

Alternatively, if you already have a rule for the servers you want to protect, edit the rule.

Step 3 Select whether to apply the rule to a specific interface or globally to all interfaces, and click **Next**.

Step 4 For Traffic Classification, select **Source and Destination IP Addresses (uses ACL)** and click **Next**.

Step 5 For the ACL rule, enter the IP addresses of the servers in **Destination**, and specify the protocol for the servers. Typically, you would use **any** for the **Source**. Click **Next** when finished.

For example, if you want to protect the web servers 10.1.1.5 and 10.1.1.6, enter:

- Source = any
- Destination = 10.1.1.5, 10.1.1.6
- Destination Protocol = tcp/http

Step 6 On the Rule Actions page, click the **Connection Settings** tab and fill in these options:

- **Embryonic Connections**—The maximum number of embryonic TCP connections per host up to 2000000. The default is **0**, which means the maximum embryonic connections are allowed. For example, you could set this to 1000.
- **Per Client Embryonic Connections**—The maximum number of simultaneous embryonic TCP connections for each client up to 2000000. When a new TCP connection is requested by a client that already has the maximum per-client number of embryonic connections open through the ASA, the ASA prevents the connection. For example, you could set this to 50.

- **TCP Syn Cookie MSS**—The server maximum segment size (MSS) for SYN-cookie generation for embryonic connections upon reaching the embryonic connections limit, from 48 to 65535 . The default is 1380. This setting is meaningful only if you configure **Embryonic Connections** or **Per Client Embryonic Connections**.

Step 7 Click **Finish** to save the rule, and **Apply** to update the device.

Step 8 Choose **Configuration** > **Firewall** > **Threat Detection**, and enable at least the **TCP Intercept** statistics under the Threat Detection Statistics group.

You can simply enable all statistics, or just enable TCP Intercept. You can also adjust the monitoring window and rates.

Step 9 Choose **Home** > **Firewall Dashboard**, and look at the **Top Ten Protected Servers under SYN Attack** dashboard to monitor the results.

Click the **Detail** button to show history sampling data. The ASA samples the number of attacks 30 times during the rate interval, so for the default 30 minute period, statistics are collected every 60 seconds.

You can clear the statistics by entering the **clear threat-detection statistics tcp-intercept** command using **Tools > Command Line Interface.**

Customize Abnormal TCP Packet Handling (TCP Maps, TCP Normalizer)

The TCP normalizer identifies abnormal packets that the ASA can act on when they are detected; for example, the ASA can allow, drop, or clear the packets. TCP normalization helps protect the ASA from attacks. TCP normalization is always enabled, but you can customize how some features behave.

To customize the TCP normalizer, first define the settings using a TCP map. Then, you can apply the map to selected traffic classes using service policies.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Objects** > **TCP Maps**.

Step 2 Do one of the following:

- Click **Add** to add a new TCP map. Enter a name for the map.
- Select a map and click **Edit**.

Step 3 In the **Queue Limit** field, enter the maximum number of out-of-order packets that can be buffered and put in order for a TCP connection, between 0 and 250 packets.

The default is 0, which means this setting is disabled and the default system queue limit is used depending on the type of traffic:

- Connections for application inspection and TCP check-retransmission have a queue limit of 3 packets. If the ASA receives a TCP packet with a different window size, then the queue limit is dynamically changed to match the advertised setting.
- For other TCP connections, out-of-order packets are passed through untouched.

If you set the Queue Limit to be 1 or above, then the number of out-of-order packets allowed for all TCP traffic matches this setting. For example, for application inspection and TCP check-retransmission traffic, any advertised settings from TCP packets are ignored in favor of the Queue Limit setting. For other TCP traffic, out-of-order packets are now buffered and put in order instead of passed through untouched.

Step 4 In the **Timeout** field, set the maximum amount of time that out-of-order packets can remain in the buffer, between 1 and 20 seconds.

If they are not put in order and passed on within the timeout period, then they are dropped. The default is 4 seconds. You cannot change the timeout for any traffic if the Queue Limit is set to 0; you need to set the limit to be 1 or above for the Timeout to take effect.

Step 5 For **Reserved Bits**, select how to handle packets that have reserved bits in the TCP header: **Clear and allow** (remove the bits before allowing the packet), **Allow only** (do not change the bits, the default), or **Drop** the packet.

Step 6 Select any of the following options:

- **Clear urgent flag**—Clears the URG flag in a packet before allowing it. The URG flag is used to indicate that the packet contains information that is of higher priority than other data within the stream. The TCP RFC is vague about the exact interpretation of the URG flag, therefore end systems handle urgent offsets in different ways, which may make the end system vulnerable to attacks.
- **Drop connection on window variation**—Drops a connection that has changed its window size unexpectedly. The window size mechanism allows TCP to advertise a large window and to subsequently advertise a much smaller window without having accepted too much data. From the TCP specification, “shrinking the window” is strongly discouraged.
- **Drop packets that exceed maximum segment size**—Drops packets that exceed the MSS set by the peer.
- **Check if transmitted data is the same as original**—Enables the retransmit data checks, which prevent inconsistent TCP retransmissions.
- **Drop packets which have past-window sequence**—Drops packets that have past-window sequence numbers, namely the sequence number of a received TCP packet is greater than the right edge of the TCP receiving window. To allow these packets, deselect this option and set the **Queue Limit** to 0 (disabling the queue limit).
- **Drop SYN Packets with data**—Drops SYN packets that contain data.
- **Enable TTL Evasion Protection**—Have the maximum TTL for a connection be determined by the TTL in the initial packet. The TTL for subsequent packets can decrease, but it cannot increase. The system will reset the TTL to the lowest previously-seen TTL for that connection. This protects against TTL evasion attacks.

  For example, an attacker can send a packet that passes policy with a very short TTL. When the TTL goes to zero, a router between the ASA and the endpoint drops the packet. It is at this point that the attacker can send a malicious packet with a long TTL that appears to the ASA to be a retransmission and is passed. To the endpoint host, however, it is the first packet that has been received by the attacker. In this case, an attacker is able to succeed without security preventing the attack.
- **Verify TCP Checksum**—Verifies the TCP checksum, dropping packets that fail verification.
- **Drop SYNACK Packets with data**—Drops TCP SYNACK packets that contain data.

- **Drop packets with invalid ACK**—Drops packets with an invalid ACK. You might see invalid ACKs in the following instances:
  - In the TCP connection SYN-ACK-received status, if the ACK number of a received TCP packet is not exactly same as the sequence number of the next TCP packet sending out, it is an invalid ACK.
  - Whenever the ACK number of a received TCP packet is greater than the sequence number of the next TCP packet sending out, it is an invalid ACK.

Note
TCP packets with an invalid ACK are automatically allowed for WAAS connections.

Step 7 (Optional.) Click the **TCP Options** tab and set the action for packets that contain TCP options.

You can clear the options before allowing the packets, allow the packets if they contain a single option of a given type, or allow the packets even if they have more than one option of a given type. The default is to allow the five named options as long as a given option appears no more than once per packet (otherwise the packet is dropped), while clearing all other options. You can also elect to drop packets that contain the MD5 or any of the numbered options. Note that if a TCP connection is inspected, all options are cleared except the MSS and selective-acknowledgment (SACK) options, regardless of your configuration.

a) Select the action for the **Selective Acknowledgement**, **TCP Timestamp**, and **Window Scale** options.

Clearing the timestamp option disables PAWS and RTT.

b) Select the action for the **MSS** (Maximum Segment Size) option.

In addition to the regular allow, allow multiple, and clear actions, you can select **Specify Maximum** and enter the maximum segment size, from 68-65535. The default TCP MSS is defined on the **Configuration** > **Firewall** > **Advanced** > **TCP Options** page.

c) Select whether you want to **Allow packets with the MD5 option**.

If you deselect the checkbox, packets that contain the MD5 option are dropped. If you select the option, you can apply the normal actions of allow, allow multiple, or clear.

d) Select the action for options by number range.

Options numbered 6-7, 9-18, and 20-255 are cleared by default. You can instead allow the options, or drop packets that contain the option. You can specify different actions for different option ranges: simply enter the lower and upper number for the range, select the action, and click **Add**. To configure an action for a single option, enter the same number for the lower and upper range.

To remove a configured range, select it and click **Delete**.

Step 8 Click **OK** and **Apply**.

You can now use the TCP map in a service policy. The map affects traffic only when applied through a service policy.

Step 9 Apply the TCP map to a traffic class using a service policy.

a) Choose **Configuration > Firewall > Service Policy Rules**.
b) Add or edit a rule. You can apply the rule globally or to an interface. For example, to customize abnormal packet handling for all traffic, create a global rule that matches any traffic. Proceed to the Rule Actions page.
c) Click the **Connection Settings** tab.

d) Choose **Use TCP Map** and select the map you created.
e) Click **Finish** or **OK**, then click **Apply**.

Bypass TCP state checks for asymmetrical routing (TCP state bypass)

TCP state bypass allows traffic to bypass stateful inspection checks on ASA devices.

Implementation considerations

If you have an asymmetrical routing environment in your network, where the outbound and inbound flow for a given connection can go through two different ASA devices, you need to implement TCP state bypass on the affected traffic.

However, TCP state bypass weakens the security of your network, so you should apply bypass on very specific, limited traffic classes.

The asymmetrical routing problem

The asymmetrical routing problem occurs when return traffic for a connection goes through a different firewall than the outbound traffic. This causes packets to be dropped because they do not match existing connection entries in the fast path table.

Traffic inspection and fast path operation

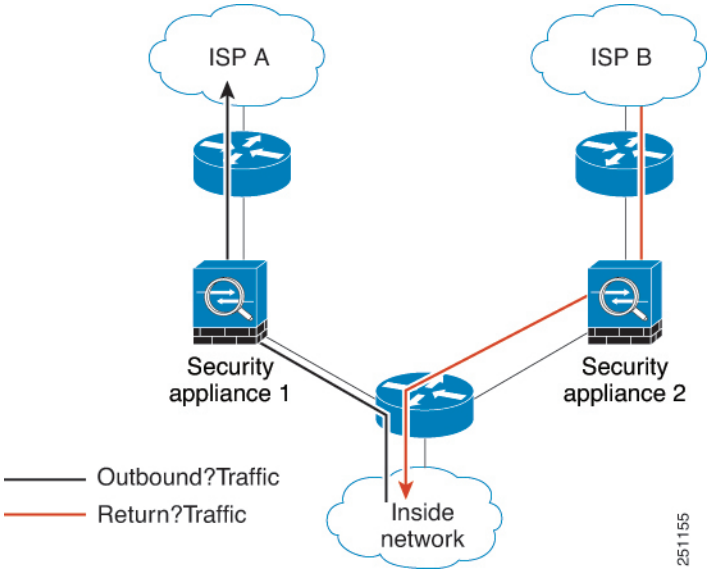
By default, the ASA inspects all traffic using the Adaptive Security Algorithm. The ASA allows or drops traffic based on the security policy. The ASA maximizes the firewall performance by checking the state of each packet (new connection or established connection). It assigns new connection SYN packets to the session management path. It assigns established connections to the fast path. It assigns packets requiring advanced inspection to the control plane path.

The ASA allows TCP packets that match existing connections to pass through via the fast path without rechecking every aspect of the security policy. This feature improves performance. However, the fast path establishment method can interfere with asymmetric routing solutions. The method uses SYN packets and performs checks such as TCP sequence number validation. Both outbound and inbound flows of a connection must pass through the same ASA device.

Asymmetric routing scenario

For example, a new connection goes to Security Appliance 1. The SYN packet goes through the session management path, and an entry for the connection is added to the fast path table. If subsequent packets of this connection go through Security Appliance 1, then the packets match the entry in the fast path, and are passed through. But if subsequent packets go to Security Appliance 2, where no SYN packet went through the session management path, then no entry exists in the fast path for the connection, and Security Appliance 2 drops the packets. The following figure shows an asymmetric routing example where the outbound traffic goes through a different ASA than the inbound traffic:

Figure 1: Asymmetric routing



If you have asymmetric routing configured on upstream routers and traffic alternates between two ASA devices, then you can configure TCP state bypass for specific traffic. TCP state bypass alters the way sessions are established in the fast path and disables the fast path checks. This feature treats TCP traffic similarly to a UDP connection: when a non-SYN packet matching the specified networks enters the ASA device, and there is not a fast path entry, then the packet goes through the session management path to establish the connection in the fast path. Once in the fast path, the traffic bypasses the fast path checks.

Guidelines and limitations for TCP state bypass

TCP state bypass unsupported features

The system does not support these features when you use TCP state bypass:

- Application inspection—Inspection requires both inbound and outbound traffic to go through the same ASA, so inspection is not applied to TCP state bypass traffic.
- AAA authenticated sessions—When a user authenticates with one ASA, traffic returning via the other ASA will be denied because the user did not authenticate with that ASA.
- TCP Intercept, maximum embryonic connection limit, TCP sequence number randomization—The ASA does not keep track of the state of the connection, so these features are not applied.
- TCP normalization—The TCP normalizer is disabled.

TCP state bypass NAT guidelines

Because the translation session is established separately for each device, be sure to configure static NAT on both devices for TCP state bypass traffic. If you use dynamic NAT, the address chosen for the session on Device 1 will differ from the address chosen for the session on Device 2.

Configure TCP State Bypass

To bypass TCP state checking in asymetrical routing environments, carefully define a traffic class that applies to the affected hosts or networks only, then enable TCP State Bypass on the traffic class using a service policy. Because bypass reduces the security of the network, limit its application as much as possible.

Before you begin

If there is no traffic on a given connection for 2 minutes, the connection times out. You can override this default by changing the **Idle Connection Timeout** for the TCP state bypass traffic class. Normal TCP connections timeout by default after 60 minutes.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Service Policy**.

Step 2 Click **Add** > **Add Service Policy Rule**.

Alternatively, if you already have a rule for the hosts, edit the rule.

Step 3 Select whether to apply the rule to a specific interface or globally to all interfaces, and click **Next**.

Step 4 For Traffic Classification, select **Source and Destination IP Addresses (uses ACL)** and click **Next**.

Step 5 For the ACL rule, enter the IP addresses of the hosts on each end of the route in **Source** and **Destination**, and specify the protocol as TCP. Click **Next** when finished.

For example, if you want to bypass TCP state checking between 10.1.1.1 and 10.2.2.2, enter:

- Source = 10.1.1.1
- Destination = 10.2.2.2
- Destination Protocol = tcp

Step 6 On the Rule Actions page, click the **Connection Settings** tab and select **TCP State Bypass**.

Step 7 Click **Finish** to save the rule, and **Apply** to update the device.

Disable TCP Sequence Randomization

Each TCP connection has two ISNs: one generated by the client and one generated by the server. The ASA randomizes the ISN of the TCP SYN passing in both the inbound and outbound directions.

Randomizing the ISN of the protected host prevents an attacker from predicting the next ISN for a new connection and potentially hijacking the new session. However, TCP sequence randomization effectively breaks TCP SACK (Selective Acknowledgement), as the sequence numbers the client sees are different from what the server sees.

You can disable TCP initial sequence number randomization if necessary, for example, because data is getting scrambled. For example:

- If another in-line firewall is also randomizing the initial sequence numbers, there is no need for both firewalls to be performing this action, even though this action does not affect the traffic.

- If you use eBGP multi-hop through the ASA, and the eBGP peers are using MD5. Randomization breaks the MD5 checksum.
- You use a WAAS device that requires the ASA not to randomize the sequence numbers of connections.
- You enable hardware bypass for the ISA 3000, and TCP connections are dropped when the ISA 3000 is no longer part of the data path.

Note We do not recommend disabling TCP sequence randomization when using clustering. There is a small chance that some TCP sessions won't be established, because the SYN/ACK packet might be dropped.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Service Policy**.

Step 2 Click **Add** > **Add Service Policy Rule**.

Alternatively, if you already have a rule for the targeted traffic, edit the rule.

Step 3 Select whether to apply the rule to a specific interface or globally to all interfaces, and click **Next**.

Step 4 For Traffic Classification, identity the type of traffic match. The class match should be for TCP traffic; you can identify specific hosts (with an ACL) do a TCP port match, or simply match any traffic. Click **Next** and configure the hosts in the ACL or define the ports, and click **Next** again.

For example, if you want to disable TCP sequence number randomization for all TCP traffic directed at 10.2.2.2, enter:

- Source = any
- Destination = 10.2.2.2
- Destination Protocol = tcp

Step 5 On the Rule Actions page, click the **Connection Settings** tab and uncheck **Randomize Sequence Number**.

Step 6 Click **Finish** to save the rule, and **Apply** to update the device.

Offload Large Flows

If you deploy the ASA on supported devices in a data center, you can identify select traffic to be offloaded to a super fast path, where traffic is switched in the NIC itself. Offloading can help you improve performance for data-intensive applications such as large file transfers.

- High Performance Computing (HPC) Research sites, where the ASA is deployed between storage and high compute stations. When one research site backs up using FTP file transfer or file sync over NFS, the large amount of data traffic affects all contexts on the ASA. Offloading FTP file transfer and file sync over NFS reduces the impact on other traffic.

- High Frequency Trading (HFT), where the ASA is deployed between workstations and the Exchange, mainly for compliance purposes. Security is usually not a concern, but latency is a major concern.

Before being offloaded, the ASA first applies normal security processing, such as access rules and inspection, during connection establishment. The ASA also does session tear-down. But once a connection is established, if it is eligible to be offloaded, further processing happens in the NIC rather than the ASA.

Offloaded flows continue to receive limited stateful inspection, such as basic TCP flag and option checking, and checksum verification if you configure it. The system can selectively escalate packets to the firewall system for further processing if necessary.

To identify flows that can be offloaded, you create a service policy rule that applies the flow offloading service. A matching flow is then offloaded if it meets the following conditions:

- IPv4 addresses only.
- TCP, UDP, GRE only.
- Standard or 802.1Q tagged Ethernet frames only.
- (Transparent mode only.) Multicast flows for bridge groups that contain two and only two interfaces.

Reverse flows for offloaded flows are also offloaded.

Restrictions for flow offload implementation

Device limitations

The feature is supported on the following devices:

- Secure Firewall 3100 (static only)
- Secure Firewall 4200 (static only)
- Secure Firewall 6100
- Firepower 4100/9300

Flows that cannot be offloaded

Do not attempt to offload these flow types:

- Any flows that do not use IPv4 addressing, such as IPv6 addressing.
- Flows for any protocol other than TCP, UDP, and GRE.

Note PPTP GRE connections cannot be offloaded.

- Flows that require inspection. In some cases, such as FTP, the secondary data channel can be offloaded although the control channel cannot be offloaded.
- IPsec and TLS/DTLS VPN connections that terminate on the device.
- Multicast flows in routed mode.

- Multicast flows in transparent mode for bridge groups that have three or more interfaces.
- TCP Intercept flows.
- TCP state bypass flows. You cannot configure flow offload and TCP state bypass on the same traffic.
- AAA cut-through proxy flows.
- Vpath, VXLAN related flows.
- Flows tagged with security groups.
- Except for the Secure Firewall 4200: Reverse flows that are forwarded from a different cluster node, in the case of asymmetric flows in a cluster.
- Centralized flows in a cluster, if the flow owner is not the control unit.

Additional limitations

Consider these operational restrictions when implementing flow offload:

- Flow offload and Dead Connection Detection (DCD) are not compatible. Do not configure DCD on connections that can be offloaded.
- If more than one flow that matches flow offload conditions are queued to be offloaded at the same time to the same location on the hardware, only the first flow is offloaded. The other flows are processed normally. This is called a *collision*. Use the **show flow-offload flow** command in the CLI to display statistics for this situation.
- Although offloaded flows pass through FXOS interfaces, statistics for these flows do not appear on the logical device interface. Thus, logical device interface counters and packet rates do not reflect offloaded flows.

Cluster redirection

When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, offloading these flows can improve performance, for example, for distribited site-to-site VPN mode. This feature is supported for these environments:

- Secure Firewall 4200
- Routed and transparent mode
- Spanned EtherChannel mode
- Multiple context mode

Conditions for reversing offload

After a flow is offloaded, packets within the flow are returned to the firewall operating system for further processing if they meet these conditions:

- They include TCP options other than Timestamp.
- They are fragmented.

- They are subject to Equal-Cost Multi-Path (ECMP) routing, and ingress packets move from one interface to another.
- A change to the routing table applies to the flow. A packet is returned to the device to determine the new route.

Configure Flow Offload

To configure flow offload, you must enable the service and then create service policies to identify the traffic that is eligible for offloading. For the Firepower 4100/9300: When you initially enable the service, you must reboot. Flow offload is enabled by default for the Secure Firewall 3100/4200.

Procedure

Step 1 (Secure Firewall 4200) Configure cluster redirect.

flow-offload cluster-redirect

Cluster redirect is enabled by default. This feature requires the **flow-offload enable** command, which is also enabled by default.

For asymmetric flows, cluster redirect lets the forwarding node offload flows to hardware. When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, letting the forwarder offload these flows can improve performance.

Step 2 Enable the flow offload service.

Flow offload is enabled by default for the Secure Firewall 3100/4200/6100.

For the Firepower 4100/9300: When you initially enable the service, you must reboot.

If a reload is required, there are special considerations for clusters or failover pairs if you want a hitless change:

- Clustering—First enter the command on the control node, but do not reboot the control node immediately. Instead, reboot each node of the cluster first, then return to the control node and reboot it. You can then configure the offloading service policy on the control node.
- Failover—First enter the command on the active unit, but do not reboot it immediately. Instead, reboot the standby unit, then reboot the active unit. You can then configure the offloading service policy on the active unit.

a) Select **Configuration** > **Firewall** > **Advanced** > **Offload Engine**.
b) Check **Enable Offload Engine**.
c) (Optional) (Secure Firewall 4200) Check **Cluster Redirect Offload**.

Cluster redirect is enabled by default.

For asymmetric flows, cluster redirect lets the forwarding node offload flows to hardware. When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, letting the forwarder offload these flows can improve performance.

d) Click **Apply**.
e) Click **Save** to save your changes to the startup configuration.

f) (Firepower 4100/9300) Select **Tools** > **System Reload** to reboot the device.

Step 3 Create the service policy rule that identifies traffic that is eligible for offload.

a) Choose **Configuration** > **Firewall** > **Service Policy**.
b) Click **Add** > **Add Service Policy Rule**.

Alternatively, if you already have a rule for the hosts, edit the rule.

c) Select whether to apply the rule to a specific interface or globally to all interfaces, and click **Next**.
d) For Traffic Classification, matching by access-list (**Source and Destination IP Addresses (uses ACL)**) or port (**TCP or UDP or SCTP Destination Port**) would be the most typical options. Select an option and click **Next**.
e) Enter the ACL or port criteria. Click **Next** when finished.

For example, if you want to make all TCP traffic on the 10.1.1.0/255.255.255.224 subnet eligible for offload, enter:

- Source = 10.1.1.0/255.255.255.224 (or 10.1.1.0/27)
- Destination = any
- Destination Protocol = tcp

f) On the Rule Actions page, click the **Connection Settings** tab and select **Flow Offload**.
g) Click **Finish** to save the rule, and **Apply** to update the device.

IPsec flow offload

IPsec flow offload is a performance optimization feature that

- offloads IPsec connections to field-programmable gate array (FPGA) or specialized hardware components after initial setup
- improves device performance by handling pre-decryption, decryption, pre-encryption, and encryption processing in hardware, and
- is enabled by default on supporting device models while allowing system software to handle inner flow security policies.

IPsec flow offload characteristics

After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the field-programmable gate array (FPGA) in the device. This process improves device performance.In the Secure Firewall 1200 series, IPsec connections are offloaded to the Marvell Cryptographic Accelerator (CPT) to improve device performance.In the Secure Firewall 6100 series, IPsec connections are offloaded to the Kintex 7 (KC400) FPGA. This FPGA contains a built-in crypto engine that is capable of handling AES-GCM-128 and AES-GCM-256 encryption and decryption.

Offloaded operations include pre-decryption and decryption processing on ingress and egress. The system software applies your security policies for traffic within the inner flow.

IPsec flow offload applies to these device types:

- Secure Firewall 1200

- Secure Firewall 3100
- Secure Firewall 4200
- Secure Firewall 6100

IPsec flow offload is also used when the device's VTI loopback interface is enabled.

For asymmetric flows in cluster distributed site-to-site VPN mode, IPsec flow offload lets the flow owner decrypt IPsec traffic in hardware that was forwarded over the cluster control link. This feature is not configurable and is always available with IPsec flow offload.

Limitations for IPsec flow offload

IPsec flows that are not offloaded include:

- IKEv1 tunnels. Only IKEv2 tunnels will be offloaded. IKEv2 supports stronger ciphers.
- Flows that have volume-based rekeying configured.
- Flows that have compression configured.
- Flows that have post-fragmentation configured.
- Flows that have anti-replay window size other than 64 bits and anti-replay is not disabled.
- AH format. Only ESP or NAT-T format will be supported.
- Flows that have firewall filter enabled.
- Modes:
  - Transport mode flows. Only tunnel mode flows will be offloaded.
  - Multiple context mode.
- Secure Firewall 6100 supports AES-GCM-128 and AES-GCM-256 ciphers only. IPsec tunnels that are configured with other ciphers are not offloaded. Any IPsec packets that are not offloaded are processed by the software engine in the CPU.

Configure IPsec Flow Offload

IPsec flow offload is enabled by default on hardware platforms that support the feature. However, egress optimization is not enabled by default, so you need to configure it if you want the feature.

Before you begin

IPsec flow offload is configured globally. You cannot configure it for selected traffic flows.

Use the **no** form of these commands to disable the features.

To see the current configuration state, use the **show flow-offload ipsec info** command.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Advanced** > **IPsec Offload**.

Step 2 Select **IPsec Offload** to enable IPsec flow offload.

Step 3 Select **Egress Optimization For IPSec Offload** to optimize the data path to enhance performance for single tunnel flows.

The configuration for egress optimization is separate from flow offload. However, even if enabled, it is effective only if you also enable IPsec flow offload.

DTLS crypto acceleration

ASA supports the Datagram Transport Layer Security (*DTLS*) cryptographic acceleration for specific firewall models. This feature improves the throughput of the DTLS-encrypted and DTLS-decrypted traffic for both IPv4 and IPv6 traffic. The ASA also performs optimization of the egress-encrypted packets to improve latency. The data path is optimized to enhance performance for single tunnel flows. Both features are enabled by default and work only for DTLS 1.2.

Supported models

- Secure Firewall 3100
- Secure Firewall 4200
- Secure Firewall 6100

Crypto accelerator details

In Secure Firewall 3100 and 4200, FPGA and the Nitrox V crypto accelerator support DTLS cryptographic acceleration

In Secure Firewall 6100, two crypto accelerators replace the Nitrox V: one for Public Key Infrastructure (PKI) acceleration and another for IPsec and DTLS offload. Only AES-GCM 128 and 256 ciphers are supported.

Configure DTLS Crypto Acceleration

By default, DTLS crypto acceleration is enabled. You can disable it if desired.

The ASA will not perform DTLS crypto acceleration under the following conditions:

- The flows use DTLS 1.0 or packet compression.
- The DTLS keys are rekeyed.
- Clustering or multiple context mode.

Procedure

Step 1 Choose **Configuration > Firewall > Advanced > DTLS Offload**.

Step 2 Check the **DTLS Offload** check box to enable DTLS crypto acceleration on the device.

Step 3 Check the **Egress Optimization for DTLS Offload** check box to enable the optimization of egress-encrypted packets and improve latency.

Monitor DTLS crypto acceleration

Use CLI commands on the threat defense device to verify and monitor DTLS crypto acceleration and optimization of egress-encrypted packets.

- To verify the status of DTLS crypto acceleration and optimization of egress-encrypted packets, use this command:

```
ciscoasa# show flow-offload-dtls info
  DTLS offload : Enabled
  Egress Optimization: Enabled
```

- To view the DTLS crypto acceleration statistics, use this command:

```
ciscoasa# show flow-offload-dtls statistics
  Packet stats of Pipe 0
  ----------------------
  Rx Packet count : 975638666
  Tx Packet count : 975638666
  Error Packet count : 0
  Drop Packet count : 0

  CAM stats of Pipe 0
  ----------------------
  Option ID Table CAM Hit Count : 1145314723
  Option ID Table CAM Miss Count : 0
  Tunnel Table CAM Hit Count : 0
  Tunnel Table CAM Miss Count : 0
  6-Tuple CAM Hit Count : 975638666
  6-Tuple CAM Miss Count : 169676057
   NOTE: The counters displayed are cumulative counters
   for all offload applications and indicates the total packets
  offloaded
```

- To view the device's Nitrox V crypto accelerator statistics for Secure Firewall 3100 and 4200, use this command:

```
ciscoasa# show crypto accelerator statistics

Crypto Accelerator Status
-------------------------
<snip>
[Offloaded SSL Input statistics, Pipe 0]
      Input packets: 290593023
      Input bytes: 147049729714
      Decrypted packets: 290593023
      Decrypted bytes: 147049729714
   [Offloaded SSL Output statistics, Pipe 0]
      Output packets: 254271808
      Output bytes: 136352952720
      Encrypted packets: 254271808
      Encrypted bytes: 136352952720
.
.
```

.

- To view crypto accelerator statistics for Secure Firewall 6100, use this command:

```
ciscoasa# show crypto accelerator statistics

Crypto Accelerator Status
-------------------------
[Capability]
<snip>
[Accelerator 0]
   Status: OK
   Software crypto engine
<snip>
[Accelerator 1]
   Status: OK
   Asymmetric Crypto Accelerator
<snip>
[Accelerator 4]
   Status: OK
   Asymmetric Crypto Accelerator
[Accelerator 5]
   Status: OK
   Offload Crypto Accelerator
      [Offloaded IPSec Input statistics, Pipe 0]
      [Offloaded IPSec Output statistics, Pipe 0]
<snip>
[Accelerator 8]
   Status: OK
   Offload Crypto Accelerator
      [Offloaded IPSec Input statistics, Pipe 3]
      [Offloaded IPSec Output statistics, Pipe 3]
```

- To view the crypto accelerator versions for Secure Firewall 6100, use this command:

```
ciscoasa# show version
<snip>
CSF6170 up 5 days 14 hours
Start-up time 12 secs
Hardware:   CSF-6170, 785423 MB RAM, CPU AMD Zen 5 2700 MHz, 2 CPUs (512 cores)
Encryption hardware device: Cisco Asymmetric Crypto Accelerator, hw version 0x00010001

                          : Cisco Asymmetric Crypto Accelerator, hw version 0x00010001

                          : Cisco Asymmetric Crypto Accelerator, hw version 0x00010001

                          : Cisco Asymmetric Crypto Accelerator, hw version 0x00010001

                          : Cisco Offload Crypto Accelerator, hw version 0x00000001
                          : Cisco Offload Crypto Accelerator, hw version 0x00000001
                          : Cisco Offload Crypto Accelerator, hw version 0x00000001
                          : Cisco Offload Crypto Accelerator, hw version 0x00000001

<snip>
```

Configure Connection Settings for Specific Traffic Classes (All Services)

You can configure different connection settings for specific traffic classes using service policies. Use service policies to:

- Customize connection limits and timeouts used to protect against DoS and SYN-flooding attacks.
- Implement Dead Connection Detection so that valid but idle connections remain alive.
- Disable TCP sequence number randomization in cases where you do not need it.
- Customize how the TCP Normalizer protects against abnormal TCP packets.
- Implement TCP State Bypass for traffic subject to asymetrical routing. Bypass traffic is not subject to inspection.
- Implement Stream Control Transmission Protocol (SCTP) State Bypass to turn off SCTP stateful inspection.
- Implement flow offload to improve performance on supported hardware platforms.
- Decrement time-to-live (TTL) on packets so that the ASA will show up on trace route output.

Note If you decrement time to live, packets with a TTL of 1 will be dropped, but a connection will be opened for the session on the assumption that the connection might contain packets with a greater TTL. Note that some packets, such as OSPF hello packets, are sent with TTL = 1, so decrementing time to live can have unexpected consequences for transparent mode ASA devices. The decrement time-to-live settings does not impact the OSPF process when ASA is operating in a routed mode.

You can configure any combination of these settings for a given traffic class, except for TCP State Bypass and TCP Normalizer customization, which are mutually exclusive.

Tip This procedure shows a service policy for traffic that goes through the ASA. You can also configure the connection maximum and embryonic connection maximum for management (to the box) traffic.

Before you begin

If you want to customize the TCP Normalizer, create the required TCP Map before proceeding.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Service Policy**, and open a rule.

- To create a new rule, click **Add > Add Service Policy Rule**. Proceed through the wizard to the Rules page.

- If you have a rule for which you are changing connection settings, select it and click **Edit**.

Step 2 On the Rule Actions wizard page or tab, select the **Connection Settings** tab.

Step 3 To set maximum connections, configure the following values in the Maximum Connections area:

By default, there are no connection limits. If you implement limits, the system must start tracking them, which can increase CPU and memory usage and result in operational problems for systems under heavy load, especially in a cluster.

- **Maximum TCP, UDP and SCTP Connections**—(TCP, UDP, SCTP.) The maximum number of simultaneous connections for all clients in the traffic class, up to 2000000. The default is 0, which means the maximum possible connections are allowed. For TCP connections, this applies to established connections only.
- **Embryonic Connections**—Specifies the maximum number of embryonic TCP connections per host up to 2000000. An embryonic connection is a connection request that has not finished the necessary handshake between source and destination. The default is **0**, which means the maximum embryonic connections are allowed. By setting a non-zero limit, you enable TCP Intercept, which protects inside systems from a DoS attack perpetrated by flooding an interface with TCP SYN packets. Also set the per-client options to protect against SYN flooding.
- **Per Client Connections**—(TCP, UDP, SCTP.) Specifies the maximum number of simultaneous connections for each client up to 2000000. When a new connection is attempted by a client that already has opened the maximum per-client number of connections, the ASA rejects the connection and drops the packet. For TCP connections, this includes established, half-open, and half-closed connections.
- **Per Client Embryonic Connections**—Specifies the maximum number of simultaneous TCP embryonic connections for each client up to 2000000. When a new TCP connection is requested by a client that already has the maximum per-client number of embryonic connections open through the ASA, the ASA prevents the connection.
- **TCP Syn Cookie MSS**—The server maximum segment size (MSS) for SYN-cookie generation for embryonic connections upon reaching the embryonic connections limit, from 48 to 65535 . The default is 1380. This setting is meaningful only if you configure **Embryonic Connections** or **Per Client Embryonic Connections**.

Step 4 To configure connection timeouts, configure the following values in the TCP Timeout area:

- **Embryonic Connection Timeout**—The idle time until an embryonic (half-open) TCP connection slot is freed. Enter 0:0:0 to disable timeout for the connection. The default is 30 seconds.
- **Half Closed Connection Timeout**—The idle timeout period until a half-closed connection is closed, between 0:5:0 (for 9.1(1) and earlier) or 0:0:30 (for 9.1(2) and later) and 1193:0:0. The default is 0:10:0. Half-closed connections are not affected by DCD. Also, the ASA does not send a reset when taking down half-closed connections.
- **Idle Connection Timeout**—The idle time until a connection slot (of *any* protocol, not just TCP) is freed. Enter 0:0:0 to disable timeout for the connection. This duration must be at least 5 minutes. The default is 1 hour.
- **Send reset to TCP endpoints before timeout**—Whether the ASA should send a TCP reset message to the endpoints of the connection before freeing the connection slot.
- **Dead Connection Detection (DCD)**—Whether to enable Dead Connection Detection (DCD). Before expiring an idle connection, the ASA probes the end hosts to determine if the connection is valid. If both

hosts respond, the connection is preserved, otherwise the connection is freed. Set the maximum number of retries (default is 5, the range is 1-255) and the retry interval, which is the period to wait after each unresponsive DCD probe before sending another probe (0:0:1 to 24:0:0, default is 0:0:15). When operating in transparent firewall mode, you must configure static routes for the endpoints. You cannot configure DCD on connections that are also offloaded, so ensure DCD and flow offload traffic classes do not overlap.Use the **show conn detail** command to track how many DCD probes have been sent by the initiator and responder.

For systems that are operating in a cluster or high-availability configuration, we recommend that you do not set the interval to less than one minute (0:1:0). If the connection needs to be moved between systems, the changes required take longer than 30 seconds, and the connection might be deleted before the change is accomplished.

Step 5 To disable randomized sequence numbers, uncheck **Randomize Sequence Number**.

Randomizing the ISN of the protected host prevents an attacker from predicting the next ISN for a new connection and potentially hijacking the new session.

Step 6 To customize TCP Normalizer behavior, check **Use TCP Map** and choose an existing TCP map from the drop-down list (if available), or add a new one by clicking **New**.

Step 7 To decrement time-to-live (TTL) on packets that match the class, check **Decrement time to live for a connection**.

Decrementing TTL is necessary for the ASA to show up in trace routes as one of the hops. You must also increase the rate limit for ICMP Unreachable messages on **Configuration > Device Management > Management Access > ICMP**.

Step 8 To enable TCP state bypass, check **TCP State Bypass**.

Step 9 To enable SCTP state bypass, check **SCTP State Bypass**.

Implement SCTP State Bypass to turn off SCTP stateful inspection. For more information, see SCTP Stateful Inspection.

Step 10 (ASA on the Firepower 4100/9300 chassis, FXOS 1.1.3 or later, only.) To enable flow offload, check **Flow Offload**.

Eligible traffic is offloaded to a super fast path, where the flows are switched in the NIC itself. You must also enable the offload service. Select **Configuration** > **Firewall** > **Advanced** > **Offload Engine**.

Step 11 Click **OK** or **Finish**.

Configure TCP Options

You can configure options to control some aspects of TCP behavior. The defaults for these settings are appropriate for most networks.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Advanced** > **TCP Options**.

Step 2 Configure per-interface TCP reset behavior.

a) Select the interface you want to change and click **Edit**.
b) Select the desired options:

- **Send reset reply for denied inbound TCP packets**. Send TCP resets for all inbound TCP sessions that attempt to transit the ASA and are denied by the ASA based on access lists or AAA settings. The ASA also sends resets for packets that are allowed by an access list or AAA, but do not belong to an existing connection and are denied by the stateful firewall. Traffic between same security level interfaces is also affected. When this option is not enabled, the ASA silently discards denied packets.
- **Send reset reply for denied outbound TCP packets**. Sends TCP resets for all outbound TCP sessions that attempt to transit the ASA and are denied by the ASA based on access lists or AAA settings. The ASA also sends resets for packets that are allowed by an access list or AAA, but do not belong to an existing connection and are denied by the stateful firewall. Traffic between same security level interfaces is also affected. When this option is not enabled, the ASA silently discards denied packets. This option is enabled by default.

c) Click **OK**.

Step 3 Configure other TCP options:

- **Send reset reply for denied outside TCP packets**. Send resets for TCP packets that terminate at the least secure interface and are denied by the ASA based on access lists or AAA settings. The ASA also sends resets for packets that are allowed by an access list or AAA, but do not belong to an existing connection and are denied by the stateful firewall. When this option is not enabled, the ASA silently discards the packets of denied connections.

  We recommend that you use this option with interface PAT. This allows the ASA to terminate the IDENT from an external SMTP or FTP server. Actively resetting these connections avoids the 30-second timeout delay.

- **Force maximum segment size for TCP connection to be X bytes**. Set the maximum TCP segment size in bytes, between 48 and any maximum number. The default value is 1380 bytes. You can disable this feature by setting bytes to 0.
- **Force minimum segment size for TCP connection to be X bytes**. Override the maximum segment size to be no less than bytes , between 48 and 65535 bytes. This feature is disabled by default (set to 0).
- **Force TCP connection to linger in TIME_WAIT state for at least 15 seconds after TCP close-down**. Forces each TCP connection to linger in a shortened TIME_WAIT state of at least 15 seconds after the final normal TCP close-down sequence. You might want to use this feature if an end host application default TCP terminating sequence is a simultaneous close.
- **TCP Maximum unprocessed segments**. Sets the maximum number of TCP unprocessed segments, from 6 to 24. The default is 6. If you find that SIP phones are not connecting to the call manager, you can try increasing the maximum number of unprocessed TCP segments.

Step 4 Click **Apply**.

Monitoring Connections

Use the following pages to monitor connections:

- **Home** > **Firewall Dashboard**, and look at the **Top Ten Protected Servers under SYN Attack** dashboard to monitor TCP Intercept. Click the **Detail** button to show history sampling data. The ASA samples the number of attacks 30 times during the rate interval, so for the default 30 minute period, statistics are collected every 60 seconds.
- **Monitoring** > **Properties** > **Connections**, to see current connections.
- **Monitoring** > **Properties** > **Connection Graphs**, to monitor performance.

In addition, you can enter the following commands using **Tools** > **Command Line Interface**.

- **show conn** [**detail**]

  Shows connection information. Detailed information uses flags to indicate special connection characteristics. For example, the "b" flag indicates traffic subject to TCP State Bypass.

  When you use the **detail** keyword, you can see information about Dead Connection Detection (DCD) probing, which shows how often the connection was probed by the initiator and responder. For example, the connection details for a DCD-enabled connection would look like the following:

```
TCP dmz: 10.5.4.11/5555 inside: 10.5.4.10/40299,
    flags UO , idle 1s, uptime 32m10s, timeout 1m0s, bytes 11828,
cluster sent/rcvd bytes 0/0, owners (0,255)
  Traffic received at interface dmz
        Locally received: 0 (0 byte/s)
  Traffic received at interface inside
        Locally received: 11828 (6 byte/s)
  Initiator: 10.5.4.10, Responder: 10.5.4.11
  DCD probes sent: Initiator 5, Responder 5
```

- **show flow-offload** {**info** [**detail**] | **cpu** | **flow** [**count** | **detail**] | **statistics**}

  Shows information about the flow offloading, including general status information, CPU usage for offloading, offloaded flow counts and details, and offloaded flow statistics.

- **show service-policy**

  Shows service policy statistics, including Dead Connection Detection (DCD) statistics.

- **show threat-detection statistics top tcp-intercept** [**all** | **detail**]

  View the top 10 protected servers under attack. The **all** keyword shows the history data of all the traced servers. The **detail** keyword shows history sampling data. The ASA samples the number of attacks 30 times during the rate interval, so for the default 30 minute period, statistics are collected every 60 seconds.

Note In the ASA configuration, embryonic connections—connection requests that have not yet completed the three-way handshake process—are closed quickly and not synchronized between the active and standby devices. This design ensures HA system efficiency and security. For this reason, there might be a difference in the number of connections on both ASAs, which is to be expected.

History for Connection Settings

Feature Name	Platform Releases	Description
TCP state bypass	8.2(1)	This feature was introduced. The following command was introduced: **set connection advanced-options tcp-state-bypass**.
Connection timeout for all protocols	8.2(2)	The idle timeout was changed to apply to all protocols, not just TCP. The following screen was modified: Configuration > Firewall > Service Policies > Rule Actions > Connection Settings.
Timeout for connections using a backup static route	8.2(5)/8.4(2)	When multiple static routes exist to a network with different metrics, the ASA uses the one with the best metric at the time of connection creation. If a better route becomes available, then this timeout lets connections be closed so a connection can be reestablished to use the better route. The default is 0 (the connection never times out). To take advantage of this feature, change the timeout to a new value. We modified the following screen: Configuration > Firewall > Advanced > Global Timeouts.
Configurable timeout for PAT xlate	8.4(3)	When a PAT xlate times out (by default after 30 seconds), and the ASA reuses the port for a new translation, some upstream routers might reject the new connection because the previous connection might still be open on the upstream device. The PAT xlate timeout is now configurable, to a value between 30 seconds and 5 minutes. We modified the following screen: Configuration > Firewall > Advanced > Global Timeouts. *This feature is not available in 8.5(1) or 8.6(1).*
Increased maximum connection limits for service policy rules	9.0(1)	The maximum number of connections for service policy rules was increased from 65535 to 2000000. We modified the following screen: Configuration > Firewall > Service Policy Rules > Connection Settings.
Decreased the half-closed timeout minimum value to 30 seconds	9.1(2)	The half-closed timeout minimum value for both the global timeout and connection timeout was lowered from 5 minutes to 30 seconds to provide better DoS protection. We modified the following screens: Configuration > Firewall > Service Policy Rules > Connection Settings; Configuration > Firewall > Advanced > Global Timeouts.

Feature Name	Platform Releases	Description
Connection holddown timeout for route convergence.	9.4(3) 9.6(2)	You can now configure how long the system should maintain a connection when the route used by the connection no longer exists or is inactive. If the route does not become active within this holddown period, the connection is freed. You can reduce the holddown timer to make route convergence happen more quickly. However, the 15 second default is appropriate for most networks to prevent route flapping. We modified the following screen: **Configuration** > **Firewall** > **Advanced** > **Global Timeouts**.
SCTP idle timeout and SCTP state bypass	9.5(2)	You can set an idle timeout for SCTP connections. You can also enable SCTP state bypass to turn off SCTP stateful inspection on a class of traffic. We modified the following screens: **Configuration** > **Firewall** > **Advanced** > **Global Timeouts**; **Configuration** > **Firewall** > **Service Policy Rules** wizard, **Connection Settings** tab.
Flow offload for the ASA on the Firepower 9300.	9.5(2.1)	You can identify flows that should be offloaded from the ASA and switched directly in the NIC (on the Firepower 9300). This provides improved performance for large data flows in data centers. This feature requires FXOS 1.1.3. We added or modified the following screens: **Configuration** > **Firewall** > **Advanced** > **Offload Engine**, the **Rule Actions** > **Connection Settings** tab when adding or editing rules under **Configuration** > **Firewall** > **Service Policy Rules**.
Flow offload support for the ASA on the Firepower 4100 series.	9.6(1)	You can identify flows that should be offloaded from the ASA and switched directly in the NIC for the Firepower 4100 series. This feature requires FXOS 1.1.4. There are no new commands or ASDM screens for this feature.
Flow offload support for multicast connections in transparent mode.	9.6(2)	You can now offload multicast connections to be switched directly in the NIC on transparent mode Firepower 4100 and 9300 series devices. Multicast offload is available for bridge groups that contain two and only two interfaces. There are no new commands or ASDM screens for this feature.

Feature Name	Platform Releases	Description
Changes in TCP option handling.	9.6(2)	You can now specify actions for the TCP MSS and MD5 options in a packet's TCP header when configuring a TCP map. In addition, the default handling of the MSS, timestamp, window-size, and selective-ack options has changed. Previously, these options were allowed, even if there were more than one option of a given type in the header. Now, packets are dropped by default if they contain more than one option of a given type. For example, previously a packet with 2 timestamp options would be allowed, now it will be dropped. You can configure a TCP map to allow multiple options of the same type for MD5, MSS, selective-ack, timestamp, and window-size. For the MD5 option, the previous default was to clear the option, whereas the default now is to allow it. You can also drop packets that contain the MD5 option. For the MSS option, you can set the maximum segment size in the TCP map (per traffic class). The default for all other TCP options remains the same: they are cleared. We modified the following screen: **Configuration** > **Firewall** > **Objects** > **TCP Maps** Add/Edit dialog box.
Stale route timeout for interior gateway protocols	9.7(1)	You can now configure the timeout for removing stale routes for interior gateway protocols such as OSPF. We modified the following screen: **Configuration** > **Firewall** > **Advanced** > **Global Timeouts**.
Global timeout for ICMP errors	9.8(1)	You can now set the idle time before the ASA removes an ICMP connection after receiving an ICMP echo-reply packet. When this timeout is disabled (the default), and you enable ICMP inspection, then the ASA removes the ICMP connection as soon as an echo-reply is received; thus any ICMP errors that are generated for the (now closed) connection are dropped. This timeout delays the removal of ICMP connections so you can receive important ICMP errors. We modified the following screen: **Configuration** > **Firewall** > **Advanced** > **Global Timeouts**.
Default idle timeout for TCP state bypass	9.10(1)	The default idle timeout for TCP state bypass connections is now 2 minutes instead of 1 hour.
Initiator and responder information for Dead Connection Detection (DCD), and DCD support in a cluster.	9.13(1)	If you enable Dead Connection Detection (DCD), you can use the **show conn detail** command to get information about the initiator and responder. Dead Connection Detection allows you to maintain an inactive connection, and the **show conn** output tells you how often the endpoints have been probed. In addition, DCD is now supported in a cluster. New/Modified screens: none.

Feature Name	Platform Releases	Description
Configure the maximum segment size (MSS) for embryonic connections.	9.16(1)	You can configure a service policy to set the server maximum segment size (MSS) for SYN-cookie generation for embryonic connections upon reaching the embryonic connections limit. This is meaningful for service policies where you are also setting embryonic connection maximums. New or changed screens: **Connection Settings** in the Add/Edit Service Policy wizard.
Configure the SIP Call Init idle global timeout	9.16(x)	You can configure the idle time after which a SIP session in Call_Init call state (after the initial invite message is received) will be deleted, i.e. when no response is received, between 0:0:1 and 1193:0:0. The default is 32 seconds (0:00:32).
IPsec flow offload.	9.18(1)	On the Secure Firewall 3100, IPsec flows are offloaded by default. After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the field-programmable gate array (FPGA) in the device, which should improve device performance. We added the following screen: **Configuration** > **Firewall** > **Advanced** > **IPsec Offload**
DTLS Crypto Acceleration	9.22(1)	Cisco Secure Firewall 4200 and 3100 series support DTLS cryptographic acceleration. The hardware performs DTLS encryption and decryption, and improves the throughput of the DTLS-encrypted and DTLS-decrypted traffic. The hardware also performs optimization of the egress-encrypted packets to improve latency. New/Modified screens: **Configuration > Firewall > Advanced > DTLS Offload > DTLS Offload** and **Egress Optimization for DTLS Offload** check boxes.
Flow offload is enabled by default for the Secure Firewall 3100/4200	9.23(1)	Flow offload is now enabled by default. Added/modified screens: **Configuration** > **Firewall** > **Advanced** > **Offload Engine**
Cluster redirect: flow offload support for the Secure Firewall 4200 asymmetric cluster traffic	9.23(1)	For asymmetric flows, cluster redirect lets the forwarding node offload flows to hardware. This feature is enabled by default. When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, letting the forwarder offload these flows can improve performance. Added/modified screens: **Configuration** > **Firewall** > **Advanced** > **Offload Engine** > **Cluster Redirect Offload**

Feature Name	Platform Releases	Description
IPsec flow offload for traffic on the cluster control link on the Secure Firewall 4200 in distributed site-to-site VPN mode	9.23(1)	For asymmetric flows in distributed site-to-site VPN mode, IPsec flow offload now lets the flow owner decrypt IPsec traffic in hardware that was forwarded over the cluster control link. This feature is not configurable and is always available when you enable IPsec flow offload. Added/modified screens: **Configuration** > **Firewall** > **Advanced** > **IPsec Offload**
IPsec flow offload, DTLS crypto acceleration, and flow offload support for the Secure Firewall 6100	9.24(1)	For IPsec and DTLS, the Secure Firewall 6100 supports AES-GCM-128 and AES-GCM-256 ciphers only.