CHAPTER 53

Configuring Connection Limits and Timeouts

This chapter describes how to set maximum TCP and UDP connections, maximum embryonic connections, maximum per-client connections, connection timeouts, dead connection detection, and how to disable TCP sequence randomization. You can set limits for connections that go through the ASA, or for management connections to the ASA. This chapter contains the following sections:

Note You can also configure maximum connections, maximum embryonic connections, and TCP sequence randomization in the NAT configuration. If you configure these settings for the same traffic using both methods, then the ASA uses the lower limit. For TCP sequence randomization, if it is disabled using either method, then the ASA disables TCP sequence randomization.

Information About Connection Limits

This section describes why you might want to limit connections, and includes the following topics:

TCP Intercept

Limiting the number of embryonic connections protects you from a DoS attack. The ASA uses the per-client limits and the embryonic connection limit to trigger TCP Intercept, which protects inside systems from a DoS attack perpetrated by flooding an interface with TCP SYN packets. An embryonic connection is a connection request that has not finished the necessary handshake between source and destination. TCP Intercept uses the SYN cookies algorithm to prevent TCP SYN-flooding attacks. A SYN-flooding attack consists of a series of SYN packets usually originating from spoofed IP addresses. The constant flood of SYN packets keeps the server SYN queue full, which prevents it from servicing connection requests. 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. When the ASA receives an ACK back from the client, it can then authenticate the client and allow the connection to the server.

To view TCP Intercept statistics, including the top 10 servers under attack, see Chapter 50, "Configuring Threat Detection."

Disabling TCP Intercept for Management Packets for Clientless SSL Compatibility

By default, TCP management connections have TCP Intercept always enabled. When TCP Intercept is enabled, it intercepts the 3-way TCP connection establishment handshake packets and thus deprives the ASA from processing the packets for clientless SSL. Clientless SSL requires the ability to process the 3-way handshake packets to provide selective ACK and other TCP options for clientless SSL connections. To disable TCP Intercept for management traffic, you can set the embryonic connection limit; only after the embryonic connection limit is reached is TCP Intercept enabled.

Dead Connection Detection (DCD)

DCD detects a dead connection and allows it to expire, without expiring connections that can still handle traffic. You configure DCD when you want idle, but valid connections to persist.

When you enable DCD, idle timeout behavior changes. With idle timeout, DCD probes are sent to each of the two end-hosts to determine the validity of the connection. If an end-host fails to respond after probes are sent at the configured intervals, the connection is freed, and reset values, if configured, are sent to each of the end-hosts. If both end-hosts respond that the connection is valid, the activity timeout is updated to the current time and the idle timeout is rescheduled accordingly.

Enabling DCD changes the behavior of idle-timeout handling in the TCP normalizer. DCD probing resets the idle timeout on the connections seen in the **show conn** command. To determine when a connection that has exceeded the configured timeout value in the timeout command but is kept alive due to DCD probing, the **show service-policy** command includes counters to show the amount of activity from DCD.

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 predecting the next ISN for a new connection and potentially hijacking the new session.

TCP initial sequence number randomization can be disabled if required. 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.

Configuring Connection Limits and Timeouts

To set connection limits and timeouts, perform the following steps.

Step 1 To identify the traffic, add a class map using the **class-map** command. See the "Creating a Layer 3/4 Class Map for Through Traffic" section on page 9-13 or the "Creating a Layer 3/4 Class Map for Management Traffic" section on page 9-15 for more information.

For example, you can match all traffic using the following commands:

```
hostname(config)# class-map CONNS
hostname(config-cmap)# match any
```

To match specific traffic, you can match an access list:

```
hostname(config)# access list CONNS extended permit ip any 10.1.1.1 255.255.255.255
hostname(config)# class-map CONNS
hostname(config-cmap)# match access-list CONNS
```

Step 2 To add or edit a policy map that sets the actions to take with the class map traffic, enter the following commands:

```
hostname(config)# policy-map name
hostname(config-pmap)# class class_map_name
hostname(config-pmap-c)#
```

where the *class_map_name* is the class map from Step 1.

For example:

```
hostname(config)# policy-map CONNS
hostname(config-pmap)# class CONNS
hostname(config-pmap-c)#
```

Step 3 To set maximum connection limits or whether TCP sequence randomization is enabled, enter the following command:

```
hostname(config-pmap-c)# set connection {[conn-max n] [embryonic-conn-max n]
[per-client-embryonic-max n] [per-client-max n] [random-sequence-number {enable |
disable}]}
```

where the **conn-max** *n* argument sets the maximum number of simultaneous TCP and/or UDP connections that are allowed, between 0 and 65535. The default is 0, which allows unlimited connections.

If two servers are configured to allow simultaneous TCP and/or UDP connections, the connection limit is applied to each configured server separately.

The **embryonic-conn-max** *n* argument sets the maximum number of simultaneous embryonic connections allowed, between 0 and 65535. The default is 0, which allows unlimited connections.

The **per-client-embryonic-max** *n* argument sets the maximum number of simultaneous embryonic connections allowed per client, between 0 and 65535. The default is 0, which allows unlimited connections.

The **per-client-max** *n* argument sets the maximum number of simultaneous connections allowed per client, between 0 and 65535. The default is 0, which allows unlimited connections.

The **random-sequence-number** {**enable** | **disable**} keyword enables or disables TCP sequence number randomization. See the "TCP Sequence Randomization" section on page 53-2 section for more information.

You can enter this command all on one line (in any order), or you can enter each attribute as a separate command. The ASA combines the command into one line in the running configuration.

Note Depending on the number of CPU cores on your ASA model, the maximum concurrent and embryonic connections may 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.

Note For management traffic, you can only set the **conn-max** and **embryonic-conn-max** keywords.

Step 4 To set connection timeouts, enter the following command:

```
hostname(config-pmap-c)# set connection timeout {[embryonic hh:mm:ss] {idle hh:mm:ss
[reset]] [half-closed hh:mm:ss] [dcd hh:mm:ss [max_retries]]}
```

where the **embryonic** *hh***:***mm:ss* keyword sets the timeout period until a TCP embryonic (half-open) connection is closed, between 0:0:5 and 1193:00:00. The default is 0:0:30. You can also set this value to 0, which means the connection never times out.

The **idle** *hh***:***mm:ss* keyword sets the idle timeout for all protocols between 0:5:0 and 1193:00:00. The default is 1:0:0. You can also set this value to 0, which means the connection never times out. For TCP traffic, the **reset** keyword sends a reset to TCP endpoints when the connection times out.

The **half-closed** *hh***:***mm:ss* keyword sets the idle timeout between 0:5:0 and 1193:00:00. 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.

The **dcd** keyword enables DCD. DCD detects a dead connection and allows it to expire, without expiring connections that can still handle traffic. You configure DCD when you want idle, but valid connections to persist. After a TCP connection times out, the ASA sends DCD probes to the end hosts to determine the validity of the connection. If one of the end hosts fails to respond after the maximum retries are exhausted, the ASA frees the connection. If both end hosts respond that the connection is valid, the ASA updates the activity timeout to the current time and reschedules the idle timeout accordingly. The *retry-interval* sets the time duration in *hh***:***mm***:***ss* format to wait after each unresponsive DCD probe before sending another probe, between 0:0:1 and 24:0:0. The default is 0:0:15. The *max-retries* sets the number of consecutive failed retries for DCD before declaring the connection as dead. The minimum value is 1 and the maximum value is 255. The default is 5.

You can enter this command all on one line (in any order), or you can enter each attribute as a separate command. The command is combined onto one line in the running configuration.

Note This command is not available for management traffic.

Step 5 To activate the policy map on one or more interfaces, enter the following command:

```
hostname(config)# service-policy policymap_name {global | interface interface_name}
```

where *policy_map_name* is the policy map you configured in Step 2. To apply the policy map to traffic on all the interfaces, use the **global** keyword. To apply the policy map to traffic on a specific interface, use the **interface** *interface_name* option, where *interface_name* is the name assigned to the interface with the **nameif** command.

Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Configuration Examples for Connection Limits and Timeouts

The following example sets the connection limits and timeouts for all traffic:

```
hostname(config)# class-map CONNS
hostname(config-cmap)# match any
hostname(config-cmap)# policy-map CONNS
hostname(config-pmap)# class CONNS
hostname(config-pmap-c)# set connection conn-max 1000 embryonic-conn-max 3000
hostname(config-pmap-c)# set connection timeout tcp 2:0:0 embryonic 0:40:0 half-closed
0:20:0 dcd
hostname(config-pmap-c)# service-policy CONNS interface outside
```

You can enter **set connection** commands with multiple parameters or you can enter each parameter as a separate command. The ASA combines the commands into one line in the running configuration. For example, if you entered the following two commands in class configuration mode:

```
hostname(config-pmap-c)# set connection conn-max 600
hostname(config-pmap-c)# set connection embryonic-conn-max 50
```

the output of the **show running-config policy-map** command would display the result of the two commands in a single, combined command:

```
set connection conn-max 600 embryonic-conn-max 50
```