CHAPTER 52

# Configuring TCP Normalization

The TCP normalization feature 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.

This chapter includes the following sections:

## Information About TCP Normalization

The TCP normalizer includes non-configurable actions and configurable actions. Typically, non-configurable actions that drop or clear connections apply to packets that are always bad. Configurable actions (as detailed in "Customizing the TCP Normalizer" section on page 52-1) might need to be customized depending on your network needs.

See the following guidelines for TCP normalization:

- The normalizer does not protect from SYN floods. The ASA includes SYN flood protection in other ways.
- The normalizer always sees the SYN packet as the first packet in a flow unless the ASA is in loose mode due to failover.

## Customizing the TCP Normalizer

This feature uses Modular Policy Framework, so that customizing TCP normalization consists of identifying traffic, specifying the TCP normalization actions, and activating TCP normalization customization on an interface. See Chapter 9, "Using Modular Policy Framework," for more information.

To customize TCP normalization, perform the following steps:

**Step 1** To specify the TCP normalization criteria that you want to look for, create a TCP map by entering the following command:

```
hostname(config)# tcp-map tcp-map-name
```

For each TCP map, you can customize one or more settings.

**Step 2** (Optional) Configure the TCP map criteria by entering one or more of the following commands (see Table 52-1). If you want to customize some settings, then the defaults are used for any commands you do not enter. The default configuration includes the following settings:

```
no check-retransmission
no checksum-verification
exceed-mss allow
queue-limit 0 timeout 4
reserved-bits allow
syn-data allow
synack-data drop
invalid-ack drop
seq-past-window drop
tcp-options range 6 7 clear
tcp-options range 9 255 clear
tcp-options selective-ack allow
tcp-options timestamp allow
tcp-options window-scale allow
ttl-evasion-protection
urgent-flag clear
window-variation allow-connection
```

*Table 52-1 tcp-map Commands*

| Command | Notes |
|---|---|
| **check-retransmission** | Prevents inconsistent TCP retransmissions. |
| **checksum-verification** | Verifies the checksum. |
| **exceed-mss** {**allow** \| **drop**} | Sets the action for packets whose data length exceeds the TCP maximum segment size.<br>(Default) The **allow** keyword allows packets whose data length exceeds the TCP maximum segment size.<br>The **drop** keyword drops packets whose data length exceeds the TCP maximum segment size. |
| **invalid-ack** {**allow** \| **drop**} | Sets the action for packets with an invalid ACK. You might see invalid ACKs in the following instances:<br>• 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.<br>• 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.<br>The **allow** keyword allows packets with an invalid ACK.<br>(Default) The **drop** keyword drops packets with an invalid ACK.<br>**Note** TCP packets with an invalid ACK are automatically allowed for WAAS connections. |

*Table 52-1 tcp-map Commands (continued)*

| Command | Notes |
|---|---|
| **queue-limit** *pkt_num* [**timeout** *seconds*] | Sets the maximum number of out-of-order packets that can be buffered and put in order for a TCP connection, between 1 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:<br>• Connections for application inspection (the **inspect** command), IPS (the **ips** command), and TCP check-retransmission (the TCP map **check-retransmission** command) 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.<br>• For other TCP connections, out-of-order packets are passed through untouched.<br>If you set the **queue-limit** command 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, IPS, 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.<br>The **timeout** *seconds* argument sets 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 *pkt_num* argument is set to 0; you need to set the limit to be 1 or above for the **timeout** keyword to take effect. |
| **reserved-bits** {**allow** \| **clear** \| **drop**} | Sets the action for reserved bits in the TCP header.<br>(Default) The **allow** keyword allows packets with the reserved bits in the TCP header.<br>The **clear** keyword clears the reserved bits in the TCP header and allows the packet.<br>The **drop** keyword drops the packet with the reserved bits in the TCP header. |
| **seq-past-window** {**allow** \| **drop**} | Sets the action for 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.<br>The **allow** keyword allows packets that have past-window sequence numbers. This action is only allowed if the **queue-limit** command is set to 0 (disabled).<br>(Default) The **drop** keyword drops packets that have past-window sequence numbers. |

*Table 52-1 tcp-map Commands (continued)*

| Command | Notes |
|---|---|
| **synack-data** {**allow** \| **drop**} | Sets the action for TCP SYNACK packets that contain data.<br>The **allow** keyword allows TCP SYNACK packets that contain data.<br>(Default) The **drop** keyword drops TCP SYNACK packets that contain data. |
| **syn-data** {**allow** \| **drop**} | Sets the action for SYN packets with data.<br>(Default) The **allow** keyword allows SYN packets with data.<br>The **drop** keyword drops SYN packets with data. |
| **tcp-options** {**selective-ack** \| **timestamp** \| **window-scale**} {**allow** \| **clear**}<br>Or<br>**tcp-options range** *lower upper* {**allow** \| **clear** \| **drop**} | Sets the action for packets with TCP options, including the selective-ack, timestamp, or window-scale TCP options.<br>(Default) The **allow** keyword allows packets with the specified option.<br>(Default for **range**) The **clear** keyword clears the option and allows the packet.<br>The **drop** keyword drops the packet with the specified option.<br>The **selective-ack** keyword sets the action for the SACK option.<br>The **timestamp** keyword sets the action for the timestamp option. Clearing the timestamp option disables PAWS and RTT.<br>The **widow-scale** keyword sets the action for the window scale mechanism option.<br>The **range** keyword specifies a range of options. The *lower* argument sets the lower end of the range as 6, 7, or 9 through 255.<br>The *upper* argument sets the upper end of the range as 6, 7, or 9 through 255. |
| **ttl-evasion-protection** | Enables the TTL evasion protection. Do not disable this command it you want to prevent attacks that attempt to evade security policy.<br>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. |

*Table 52-1 tcp-map Commands (continued)*

| Command | Notes |
|---|---|
| **urgent-flag** {**allow** \| **clear**} | Sets the action for packets with the URG flag. 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.<br>The **allow** keyword allows packets with the URG flag.<br>(Default) The **clear** keyword clears the URG flag and allows the packet. |
| **window-variation** {**allow** \| **drop**} | Sets the action for 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. When this condition is detected, the connection can be dropped.<br>(Default) The **allow** keyword allows connections with a window variation.<br>The **drop** keyword drops connections with a window variation. |

**Step 3** 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 for more information.

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

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

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

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

**Step 4** 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 3.

For example:

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

**Step 5** Apply the TCP map to the class map by entering the following command.

```
hostname(config-pmap-c)# set connection advanced-options tcp-map-name
```

**Step 6** 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 **global** applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. Interface service policies take precedence over the global service policy for a given feature. For example, if you have a global policy with inspections, and an interface policy with TCP normalization, then both inspections and TCP normalization are applied to the interface. However, if you have a global policy with inspections, and an interface policy with inspections, then only the interface policy inspections are applied to that interface.

# Configuration Examples for TCP Normalization

For example, to allow urgent flag and urgent offset packets for all traffic sent to the range of TCP ports between the well known FTP data port and the Telnet port, enter the following commands:

```
hostname(config)# tcp-map tmap
hostname(config-tcp-map)# urgent-flag allow
hostname(config-tcp-map)# class-map urg-class
hostname(config-cmap)# match port tcp range ftp-data telnet
hostname(config-cmap)# policy-map pmap
hostname(config-pmap)# class urg-class
hostname(config-pmap-c)# set connection advanced-options tmap
hostname(config-pmap-c)# service-policy pmap global
```