



CHAPTER 19

Configuring Advanced Connection Features

This chapter describes how to customize connection features, and includes the following sections:

- [Configuring Connection Limits and Timeouts, page 19-1](#)
- [Configuring TCP State Bypass, page 19-4](#)
- [Preventing IP Spoofing, page 19-7](#)
- [Configuring the Fragment Size, page 19-8](#)
- [Blocking Unwanted Connections, page 19-8](#)

Configuring Connection Limits and Timeouts

This section describes how to set maximum TCP and UDP connections, connection timeouts, and how to disable TCP sequence randomization.

Each TCP connection has two ISNs: one generated by the client and one generated by the server. The FWSM 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.

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 FWSM, and the eBGP peers are using MD5. Randomization breaks the MD5 checksum.
- You use a WAAS device that requires the FWSM not to randomize the sequence numbers of connections.



Note

Because of the way TCP sequence randomization is implemented, if you enable Xlate Bypass (see the [“Configuring Xlate Bypass” section on page 12-18](#)), then disabling TCP sequence randomization only works for control connections, and not data connections; for data connections, the TCP sequence continues to be randomized.

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

NAT also lets you configure embryonic connection limits, which triggers TCP Intercept to prevent a DoS attack. To configure connection limits, TCP randomization, and embryonic limits, see [Chapter 12, “Configuring NAT.”](#)

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 [“Identifying Traffic Using a Class Map”](#) section on page 18-2 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
```



- Note** When you identify a **match access-list** command for the class map, then the **set connection** actions are performed separately for each ACE in the access list and not for the access list as a whole. For example, you match an access list with 2 ACEs such as the following, and apply a connection limit of 2 connections:

```
access-list testACL extended permit tcp host 10.2.1.1 any eq 21
access-list testACL extended permit tcp host 10.2.1.1 any eq 23

class-map testclass
  match access-list testACL

policy-map testpolicy
  class testclass
    set connection conn-max 2
```

The FWSM allows the creation of 2 connections for Telnet sessions (ACE 1) and 2 connections for FTP sessions (ACE 2).

- 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 ] [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 means no limit on connections.

The **random-sequence-number** {**enable** | **disable**} keyword enables or disables TCP sequence number randomization.

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

- Step 4** To set the timeout for TCP embryonic connections (half-opened) or TCP half-closed connections, enter the following command:

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

embryonic *hh:mm:ss*

half-closed *hh:mm:ss*



example, you cannot change the connection settings for secondary flows like SQL*Net, FTP data flows, and so on using the **set connection timeout** command. For these connections, use the global **timeout conn** command to change the idle time. Note that the **timeout conn** command affects *all* traffic flows unless you otherwise use the **set connection timeout** command for eligible traffic.

- Step 5** To set the timeout for idle connections for all protocols, enter the following command:

```
hostname(config-pmap-c)# set connection timeout idle : :0
```

where the **idle** *hh:mm:0* argument defines the idle time after which an established connection of any protocol closes, between 0:5:0 and 1092:15:0. The default is 0:60:0. You can also set the value to 0, which means the connection never times out.



Note This command ignores the value you set for seconds; you can only specify the hours and minutes. Therefore, you should set the seconds to be 0.

The **idle** keyword has replaced the **tcp** keyword in the **set connection timeout** command, but if your configuration includes the **tcp** command (for TCP connections only), it is still accepted. If your policy includes both the **idle** and **tcp** commands, then the **tcp** command takes precedence for TCP traffic only if the class map matches an access list that specifies TCP traffic explicitly. See the **set connection timeout** command in the *Catalyst 6500 Series Switch and Cisco 7600 Series Router Firewall Services Module Command Reference* for more information.

This command does not affect secondary connections created by an inspection engine. For example, you cannot change the connection settings for secondary flows like SQL*Net, FTP data flows, and so on using the **set connection timeout** command. For these connections, use the global **timeout conn** command to change the idle time. Note that the **timeout conn** command affects *all* traffic flows unless you otherwise use the **set connection timeout** command for eligible traffic.

Step 6 To activate the policy map on one or more interfaces, enter the following command:

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

where *polycmap_name* is the policy map you configured in [Step 2](#). To apply the policy map to traffic on all the interfaces, use the **global**

```
interface  
nameif
```

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.

The following example sets the maximum TCP and UDP connections to 5000, and sets the maximum embryonic timeout to 40 seconds, the half-closed timeout to 20 minutes, and the idle timeout to 2 hours for traffic going to 10.1.1.1:

```
hostname(config)# access-list CONNS permit ip any host 10.1.1.1

hostname(config)# class-map conns
hostname(config-cmap)# match access-list CONNS

hostname(config-cmap)# policy-map conns
hostname(config-pmap)# class conns
hostname(config-pmap-c)# set connection conn-max 5000
hostname(config-pmap-c)# set connection timeout embryonic 0:0:40 half-closed 0:20:0
hostname(config-pmap-c)# set connection timeout idle 2:0:0

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 FWSM 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 timeout embryonic 0:0:40
hostname(config-pmap-c)# set connection timeout half-closed 0:20:0
```

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 timeout embryonic 0:0:40 half-closed 0:20:0
```

Configuring TCP State Bypass

This section tells how to configure TCP state bypass, and includes the following topics:

- [TCP State Bypass Overview, page 19-5](#)
- [Configuring TCP State Bypass, page 19-6](#)

TCP State Bypass Overview

-
-
-
-

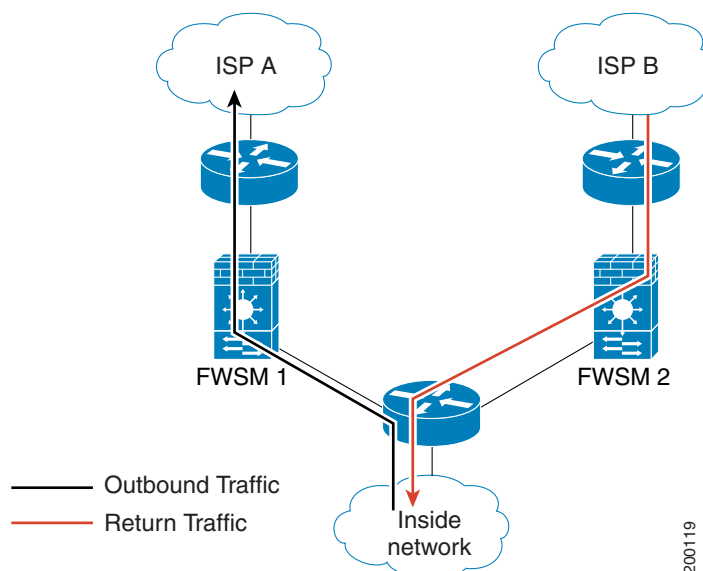
Allowing Outbound and Inbound Flows through Separate FWSMs

connection?) and assigning it to either the session management path (a new connection SYN packet), the fast path (an established connection), or the control plane path (advanced inspection). See the “[Stateful Inspection Overview](#)” section on page 1-8 for more detailed information about the stateful firewall.

TCP packets that match existing connections in the fast path can pass through the FWSM without rechecking every aspect of the security policy. This feature maximizes performance. However, the method of establishing the session in the fast path using the SYN packet, and the checks that occur in the fast path (such as TCP sequence number), can stand in the way of asymmetrical routing solutions: both the outbound and inbound flow of a connection must pass through the same FWSM.

For example, a new connection goes to FWSM 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 FWSM 1, then the packets will match the entry in the fast path, and are passed through. But if subsequent packets go to FWSM 2, where there was not a SYN packet that went through the session management path, then there is no entry in the fast path for the connection, and the packets are dropped. [Figure 19-1](#) shows an asymmetric routing example where the outbound traffic goes through a different FWSM than the inbound traffic:

Figure 19-1 Asymmetric Routing



Unsupported Features

- Application inspection—Application inspection requires both inbound and outbound traffic to go through the same FWSM, so application inspection is not supported with TCP state bypass.
- AAA authenticated sessions—When a user authenticates with one FWSM, traffic returning via the other FWSM will be denied because the user did not authenticate with that FWSM.

Compatibility with NAT

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

Connection Timeout

If there is no traffic on a given connection for 2 minutes, the connection times out. You can override this default using the **set connection timeout tcp** command. Normal TCP connections timeout by default after 60 minutes.

Configuring TCP State Bypass

To enable TCP state bypass, perform the following steps:

-
- Step 1** To identify the traffic for which you want to disable stateful firewall inspection, add a class map using the **class-map** command. See the [“Identifying Traffic Using a Class Map”](#) section on page 18-2 for more information.

For example, you can match an access list:

```
access list bypass extended permit tcp any 10.1.1.1 255.255.255.255
class-map bypass_traffic
  match access-list bypass
```

- Step 2** Add a policy map, which associates security-related actions with the class map traffic, by entering the following command:

```
policy-map
```

- Step 3** Assign the class map you created in Step 1 to the policy map you created in Step 2 by entering the following command:

```
class
```

- Step 4** Enable TCP state bypass by entering the following command:

```
set connection advanced-options tcp-state-bypass
```

Step 5 Activate the policy map on one or more interfaces by entering the following command:

```
service-policy global interface
```

Where **global** applies the policy map to all interfaces, and **interface** applies the policy to one interface. 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.



Note

If you use the **show conn** command, the display for connections that use TCP state bypass includes the flag “b.”

The following is an example configuration for TCP state bypass:

```
access-list tcp_extended permit tcp 10.1.1.0 255.255.255.0 10.2.1.0
255.255.255.0

class-map tcp_bypass
description "TCP traffic that bypasses stateful firewall"
match access-list tcp_bypass

policy-map tcp_bypass_policy
class tcp_bypass
set connection advanced-options tcp-state-bypass

service-policy tcp_bypass_policy outside
```

Preventing IP Spoofing

This section lets you enable Unicast Reverse Path Forwarding on an interface. Unicast RPF guards against IP spoofing (a packet uses an incorrect source IP address to obscure its true source) by ensuring that all packets have a source IP address that matches the correct source interface according to the routing table.

Normally, the FWSM only looks at the destination address when determining where to forward the packet. Unicast RPF instructs the FWSM to also look at the source address; this is why it is called Reverse Path Forwarding. For any traffic that you want to allow through the FWSM, the FWSM routing table must include a route back to the source address. See RFC 2267 for more information.

For outside traffic, for example, the FWSM can use the default route to satisfy the Unicast RPF protection. If traffic enters from an outside interface, and the source address is not known to the routing table, the FWSM uses the default route to correctly identify the outside interface as the source interface.

If traffic enters the outside interface from an address that is known to the routing table, but is associated with the inside interface, then the FWSM drops the packet. Similarly, if traffic enters the inside interface from an unknown source address, the FWSM drops the packet because the matching route (the default route) indicates the outside interface.

Unicast RPF is implemented as follows:

- ICMP packets have no session, so each packet is checked.

- UDP and TCP have sessions, so the initial packet requires a reverse route lookup. Subsequent packets arriving during the session are checked using an existing state maintained as part of the session. Non-initial packets are checked to ensure they arrived on the same interface used by the initial packet.

To enable Unicast RPF, enter the following command:

```
ip verify reverse-path interface
```

Configuring the Fragment Size

By default, the FWSM allows up to 24 fragments per IP packet, and up to 200 fragments awaiting reassembly. You might need to let fragments on your network if you have an application that routinely fragments packets, such as NFS over UDP. However, if you do not have an application that fragments traffic, we recommend that you do not allow fragments through the FWSM. Fragmented packets are often used as DoS attacks. To set disallow fragments, enter the following command:

```
fragment chain 1
```

Enter an interface name if you want to prevent fragmentation on a specific interface. By default, this command applies to all interfaces.

Blocking Unwanted Connections

If you know that a host is attempting to attack your network (for example, system log messages show an attack), then you can block (or shun) connections based on the source IP address and other identifying parameters. No new connections can be made until you remove the shun.



Note

If you have an IPS that monitors traffic, then the IPS can shun connections automatically.

To shun a connection manually, perform the following steps:

- Step 1** If necessary, view information about the connection by entering the following command:

```
show conn
```

The FWSM shows information about each connection, such as the following:

```
TCP out 64.101.68.161:4300 in 10.86.194.60:23 idle 0:00:00 bytes 1297 flags UIO
```

- Step 2** To shun connections from the source IP address, enter the following command:

```
hostname(config)# shun [dst_ip src_port dest_port [protocol]] [vlan vlan_id]
```

This command drops an existing connection, as well as blocking future connections. By default, the protocol is 0 for IP.

For multiple context mode, you can enter this command in the admin context, and by specifying a VLAN ID that is assigned to an interface in other contexts, you can shun the connection in other contexts.

To remove the shun, enter the following command:

```
hostname(config)# no shun src_ip [vlan vlan_id]
```