
# Preventing Network Attacks

This chapter describes how to prevent network attacks by configuring TCP normalization, limiting TCP and UDP connections, and many other protection features.

This chapter includes the following sections:



## Configuring TCP Normalization

The TCP normalization feature lets you specify criteria that identify abnormal packets, which the security appliance drops when they are detected. This feature uses Modular Policy Framework, so that implementing TCP normalization consists of identifying traffic, specifying the TCP normalization criteria, and activating TCP normalization on an interface. See Chapter 18, "Using Modular Policy Framework," for more information.

To configure 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 specify one or more settings.

**Step 2** Configure the TCP map criteria by entering commands for one or more of the following options:

- Prevent inconsistent TCP retransmissions:

  ```
  hostname(config-tcp-map)# check-retransmission
  ```

- Verify the checksum:

  ```
  hostname(config-tcp-map)# checksum-verification
  ```

- Allow packets whose data length exceeds the TCP maximum segment size. The default is to drop these packets, so use this command to allow them.

  ```
  hostname(config-tcp-map)# exceed-mss {allow | drop}
  ```

- Set the maximum number of out-of-order packets that can be queued for a TCP connection:

  ```
  hostname(config-tcp-map)# queue-limit pkt_num
  ```

  Where *pkt_num* specifies the maximum number of out-of-order packets. The range is 0 to 250 and the default is 0.

- Clear reserved bits in the TCP header, or drop packets with reserved bits set. The default is to allow reserved bits, so use this command to clear them or drop the packets.

  ```
  hostname(config-tcp-map)# reserved-bits {allow | clear | drop}
  ```

  Where **allow** allows packets with the reserved bits in the TCP header. **clear** clears the reserved bits in the TCP header and allows the packet. **drop** drops the packet with the reserved bits in the TCP header.

- Drop SYN packets with data. The default is to allow SYN packets with data, so use this command to drop the packets.

  ```
  hostname(config-tcp-map)# syn-data {allow | drop}
  ```

- Clears the selective-ack, timestamps, or window-scale TCP options, or drops a range of TCP options by number. The default is to allow packets with specified options, or to clear the options within the range, so use this command to clear, allow, or drop them.

  ```
  hostname(config-tcp-map)# tcp-options {selective-ack | timestamp | window-scale}
  {allow | clear}
  ```

  Or:

  ```
  hostname(config-tcp-map)# tcp-options range lower upper {allow | clear | drop}
  ```

  Where **allow** allows packets with the specified option. **clear** clears the option and allows the packet. **drop** drops the packet.

  The **selective-ack** keyword allows or clears the SACK option. The default is to allow the SACK option.

  The **timestamp** keyword allows or clears the timestamp option. Clearing the timestamp option disables PAWS and RTT. The default is to allow the timestamp option.

  The **widow-scale** keyword allows or clears the window scale mechanism option. The default is to allow the window scale mechanism option.

  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.

  The *upper* argument sets the upper end of the range as 6, 7, or 9 through 255.

- Disable the TTL evasion protection:

  ```
  hostname(config-tcp-map)# ttl-evasion-protection
  ```

  Do not enter this command it you want to prevent attacks that attempt to evade security policy.

  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 security appliance 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 security

appliance 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.

- Allow the URG pointer:

  ```
  hostname(config-tcp-map)# urgent-flag {allow | clear}
  ```

  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. The default behavior is to clear the URG flag and offset. Use this command to allow the URB flag.

- Drop a connection that has changed its window size unexpectedly. The default is to allow connections, so use this command to drop them.

  ```
  hostname(config-tcp-map)# window-variation {allow | drop}
  ```

  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.

**Step 3** To identify the traffic to which you want to apply TCP normalization, 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.

**Step 4** To add or edit a policy map that sets the actions to take with the class map traffic, enter the following command:

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

**Step 5** To identify the class map from Step 1 to which you want to assign an action, enter the following command:

```
hostname(config-pmap)# class class_map_name
```

**Step 6** 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 7** 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. 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.

---

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
```

# Configuring Connection Limits and Timeouts

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

Limiting the number of connections and embryonic connections protects you from a DoS attack. The security appliance 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 sequence randomization should only be disabled if another in-line firewall is also randomizing sequence numbers and the result is scrambling the data. Each TCP connection has two Initial Sequence Numbers (ISNs): one generated by the client and one generated by the server. The security appliance randomizes the ISN that is generated by the host/server. At least one of the ISNs must be randomly generated so that attackers cannot predict the next ISN and potentially hijack the session.

**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 security appliance uses the lower limit. For TCP sequence randomization, if it is disabled using either method, then the security appliance disables TCP sequence randomization.

To set connection limits, 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.

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

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

**Step 3** To identify the class map from Step 1 to which you want to assign an action, enter the following command:

```
hostname(config-pmap)# class class_map_name
```

**Step 4** To set maximum connection limits or whether TCP sequence randomization is enabled, enter the following command:

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

where *number* is an integer between 0 and 65535. The default is 0, which means no limit on connections.

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

**Step 5** To set the timeout for connections, embryonic connections (half-opened), and half-closed connections, enter the following command:

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

where **embryonic** *hh*[**:***mm*[**:***ss*] is a time between 0:0:5 and 1192:59:59. The default is 0:0:30. You can also set this value to 0, which means the connection never times out.

The **half-closed** *hh*[**:***mm*[**:***ss*] and **tcp** *hh*[**:***mm*[**:***ss*] values are a time between 0:5:0 and 1192:59:59. The default for **half-closed** is 0:10:0 and the default for **tcp** is 1:0:0. You can also set these values to 0, which means the connection never times out.

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.

**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. 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.

# 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 security appliance only looks at the destination address when determining where to forward the packet. Unicast RPF instructs the security appliance 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 security appliance, the security appliance routing table must include a route back to the source address. See RFC 2267 for more information.

For outside traffic, for example, the security appliance 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 security appliance 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 security appliance drops the packet. Similarly, if traffic enters the inside interface from an unknown source address, the security appliance 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:

```
hostname(config)# ip verify reverse-path interface interface_name
```

# Configuring the Fragment Size

By default, the security appliance 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 security appliance. Fragmented packets are often used as DoS attacks. To set disallow fragments, enter the following command:

```
hostname(config)# fragment chain 1 [interface_name]
```

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, such as an AIP SSM, 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:

```
hostname# show conn
```

The security appliance 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 src_ip [dst_ip src_port dest_port [protocol]] [vlan vlan_id]
```

If you enter only the source IP address, then all future connections are shunned; existing connections remain active.

To drop an existing connection, as well as blocking future connections from the source IP address, enter the destination IP address, source and destination ports, and the protocol. 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.

**Step 3** To remove the shun, enter the following command:

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

# Configuring IP Audit for Basic IPS Support

The IP audit feature provides basic IPS support for a security appliance that does not have an AIP SSM. It supports a basic list of signatures, and you can configure the security appliance to perform one or more actions on traffic that matches a signature.

To enable IP audit, perform the following steps:

**Step 1** To define an IP audit policy for informational signatures, enter the following command:

```
hostname(config)# ip audit name name info [action [alarm] [drop] [reset]]
```

Where **alarm** generates a system message showing that a packet matched a signature, **drop** drops the packet, and **reset** drops the packet and closes the connection. If you do not define an action, then the default action is to generate an alarm.

**Step 2** To define an IP audit policy for attack signatures, enter the following command:

```
hostname(config)# ip audit name name attack [action [alarm] [drop] [reset]]
```

Where **alarm** generates a system message showing that a packet matched a signature, **drop** drops the packet, and **reset** drops the packet and closes the connection. If you do not define an action, then the default action is to generate an alarm.

**Step 3** To assign the policy to an interface, enter the following command:

```
ip audit interface interface_name policy_name
```

**Step 4** To disable signatures, or for more information about signatures, see the **ip audit signature** command in the *Cisco Security Appliance Command Reference*.