



# Transport and Network Layer Preprocessors

The following topics explain transport and network layer preprocessors and how to configure them:

- [Introduction to Transport and Network Layer Preprocessors, on page 1](#)
- [License Requirements for Transport and Network Layer Preprocessors, on page 1](#)
- [Requirements and Prerequisites for Transport and Network Layer Preprocessors, on page 2](#)
- [Advanced Transport/Network Preprocessor Settings, on page 2](#)
- [Checksum Verification, on page 5](#)
- [The Inline Normalization Preprocessor, on page 7](#)
- [The IP Defragmentation Preprocessor, on page 13](#)
- [The Packet Decoder, on page 18](#)
- [TCP Stream Preprocessing, on page 23](#)
- [UDP Stream Preprocessing, on page 33](#)

## Introduction to Transport and Network Layer Preprocessors

Transport and network layer preprocessors detect attacks that exploit IP fragmentation, checksum validation, and TCP and UDP session preprocessing. Before packets are sent to preprocessors, the packet decoder converts packet headers and payloads into a format that can be easily used by the preprocessors and the intrusion rules engine and detects various anomalous behaviors in packet headers. After packet decoding and before sending packets to other preprocessors, the inline normalization preprocessor normalizes traffic for inline deployments.

When an intrusion rule or rule argument requires a disabled preprocessor, the system automatically uses it with its current configuration even though it remains disabled in the network analysis policy's web interface.

## License Requirements for Transport and Network Layer Preprocessors

### Threat Defense License

IPS

### Classic License

Protection

# Requirements and Prerequisites for Transport and Network Layer Preprocessors

## Model Support

Any.

## Supported Domains

Any

## User Roles

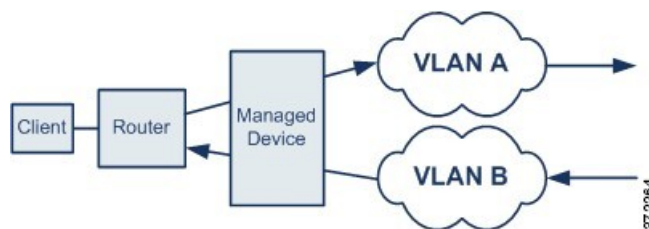
- Admin
- Intrusion Admin

## Advanced Transport/Network Preprocessor Settings

Advanced transport and network preprocessor settings apply globally to all networks, zones, and VLANs where you deploy your access control policy. You configure these advanced settings in an access control policy rather than in a network analysis policy.

## Ignored VLAN Headers

Different VLAN tags in traffic traveling in different directions for the same connection can affect traffic reassembly and rule processing. For example, in the following graphic traffic for the same connection could be transmitted over VLAN A and received over VLAN B.



You can configure the system to ignore the VLAN header so packets can be correctly processed for your deployment.

## Active Responses in Intrusion Drop Rules

A drop rule is an intrusion or preprocessor rule whose rule state is set to Drop and Generate Events. In an inline deployment, the system responds to TCP or UDP drop rules by dropping the triggering packet and blocking the session where the packet originated.



**Tip** Because UDP data streams are not typically thought of in terms of *sessions*, the stream preprocessor uses the source and destination IP address fields in the encapsulating IP datagram header and the port fields in the UDP header to determine the direction of flow and identify a UDP session.

You can configure the system to initiate one or more *active responses* to more precisely and specifically close a TCP connection or UDP session when an offending packet triggers a TCP or UDP drop rule. You can use active responses in inline, including routed and transparent, deployments. Active responses are not suited or supported for passive deployments.

To configure active responses:

- Create or modify a TCP or UDP (**resp** keyword only) intrusion rule. See [Intrusion Rule Header Protocol](#).
- Add the **react** or **resp** keyword to the intrusion rule; see [xActive Response Keywords](#).
- Optionally, for a TCP connection, specify the maximum number of additional active responses to send and the number of seconds to wait between active responses; see **Maximum Active Responses** and **Minimum Response Seconds** in [Advanced Transport/Network Preprocessor Options, on page 3](#).

Active responses close the session when matching traffic triggers a drop rule, as follows:

- **TCP**—drops the triggering packet and inserts a TCP Reset (RST) packet in both the client and server traffic.
- **UDP**—sends an ICMP unreachable packet to each end of the session.

## Advanced Transport/Network Preprocessor Options

### Ignore the VLAN header when tracking connections

Specifies whether to ignore or include VLAN headers when identifying traffic, as follows:

- When this option is selected, the system ignores VLAN headers. Use this setting for deployed devices that might detect different VLAN tags for the same connection in traffic traveling in different directions
- When this option is disabled, the system includes VLAN headers. Use this setting for deployed devices that will not detect different VLAN tags for the same connection traffic traveling in different directions.

### Maximum Active Responses

Specifies a maximum number of active responses per TCP connection. When additional traffic occurs on a connection where an active response has been initiated, and the traffic occurs more than **Minimum Response Seconds** after a previous active response, the system sends another active response unless the specified maximum has been reached. A setting of 0 disables additional active responses triggered by **resp** or **react** rules. See [Active Responses in Intrusion Drop Rules, on page 2](#) and [Active Response Keywords](#).

Note that a triggered **resp** or **react** rule initiates an active response regardless of the configuration of this option.

**Minimum Response Seconds**

Until **Maximum Active Responses** occur, specifies the number of seconds to wait before any additional traffic on a connection where the system has initiated an active response results in a subsequent active response.

**Troubleshooting Options: Session Termination Logging Threshold**


---

**Caution** Do not modify Session Termination Logging Threshold unless instructed to do so by Support.

---

Support might ask you during a troubleshooting call to configure your system to log a message when an individual connection exceeds the specified threshold. Changing the setting for this option will affect performance and should be done only with Support guidance.

This option specifies for the number of bytes that result in a logged message when the session terminates and the specified number was exceeded.




---

**Note** The upper limit of 1GB is also restricted by the amount of memory on the managed device allocated for stream processing.

---

**Related Topics**

[Active Response Keywords](#)

## Configuring Advanced Transport/Network Preprocessor Settings

You must be an Admin, Access Admin, or Network Admin to perform this task.

**Procedure**

- 
- Step 1** In the access control policy editor, click **Edit** (✎) on the policy you want to modify.
- Step 2** Click **More > Advanced Settings**, and then click **Edit** (✎) next to the **Transport/Network Preprocessor Settings** section.
- Step 3** Except for the troubleshooting option **Session Termination Logging Threshold**, modify the options described in [Advanced Transport/Network Preprocessor Options, on page 3](#).

**Caution**

Do not modify **Session Termination Logging Threshold** unless instructed to do so by Support.

- Step 4** Click **OK**.
- 

**What to do next**

- Optionally, further configure the policy as described in [Editing an access control policy](#).
- Deploy configuration changes.

# Checksum Verification



---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

The system can verify all protocol-level checksums to ensure that complete IP, TCP, UDP, and ICMP transmissions are received and that, at a basic level, packets have not been tampered with or accidentally altered in transit. A checksum uses an algorithm to verify the integrity of a protocol in the packet. The packet is considered to be unchanged if the system computes the same value that is written in the packet by the end host.

Disabling checksum verification may leave your network susceptible to insertion attacks. Note that the system does not generate checksum verification events. In an inline deployment, you can configure the system to drop packets with invalid checksums.

## Checksum Verification Options

You can set any of the following options to **Enabled** or **Disabled** in a passive or inline deployment, or to **Drop** in an inline deployment:

- **ICMP Checksums**
- **IP Checksums**
- **TCP Checksums**
- **UDP Checksums**

To drop offending packets, in addition to setting an option to **Drop** you must also enable **Inline Mode** in the associated network analysis policy and ensure that the device is deployed inline.

Setting these options to **Drop** in a passive deployment, or in an inline deployment in tap mode, is the same as setting them to **Enabled**.



---

**Attention** Under **TCP checksums**, the **Ignore** option (which is the default) bypasses or ignores any configured Snort rules.

---

The default for all checksum verification options is **Enabled**. However, Firewall Threat Defense routed and transparent interfaces always drop packets that fail IP checksum verification. Note that the Firewall Threat Defense routed and transparent interfaces fix UDP packets with a bad checksum before passing the packets to the Snort process.

### Related Topics

[Preprocessor Traffic Modification in Inline Deployments](#)

# Verifying Checksums



---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

## Procedure

---

**Step 1** Choose **Policies > Security policies > Access Control**, and then click **Network Analysis Policy** or **Policies > Security policies > Intrusion**, and then click **Network Analysis Policies**.

**Note**

If your custom user role limits access to the first path listed here, use the second path to access the policy.

**Step 2** Click **Snort 2 Version** next to the policy you want to edit.

**Step 3** Click **Edit** (✎) next to the policy you want to edit.

If **View** (👁) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

**Step 4** Click **Settings** in the navigation panel.

**Step 5** If **Checksum Verification** under **Transport/Network Layer Preprocessors** is disabled, click **Enabled**.

**Step 6** Click **Edit** (✎) next to **Checksum Verification**.

**Step 7** Modify the options described in [Checksum Verification, on page 5](#).

**Note**

Under **TCP checksums**, the **Ignore** option (which is the default) bypasses or ignores any configured Snort rules.

**Step 8** To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

---

## What to do next

- Deploy configuration changes.

## Related Topics

[Layer Management](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

# The Inline Normalization Preprocessor



---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

The inline normalization preprocessor normalizes traffic to minimize the chances of attackers evading detection in inline deployments.



---

**Note** For the system to affect traffic, you must deploy relevant configurations to managed devices using routed, switched, or transparent interfaces, or inline interface pairs.

---

You can specify normalization of any combination of IPv4, IPv6, ICMPv4, ICMPv6, and TCP traffic. Most normalizations are on a per-packet basis and are conducted by the inline normalization preprocessor. However, the TCP stream preprocessor handles most state-related packet and stream normalizations, including TCP payload normalization.

Inline normalization takes place immediately after decoding by the packet decoder and before processing by other preprocessors. Normalization proceeds from the inner to outer packet layers.

The inline normalization preprocessor does not generate events; it prepares packets for use by other preprocessors and the rules engine in inline deployments. The preprocessor also helps ensure that the packets the system processes are the same as the packets received by the hosts on your network.



---

**Note** In an inline deployment, we recommend that you enable inline mode and configure the inline normalization preprocessor with the **Normalize TCP Payload** option enabled. In a passive deployment, we recommend that you use adaptive profile updates.

---

## Related Topics

[Preprocessor Traffic Modification in Inline Deployments](#)  
[About Adaptive Profiles](#)

## Inline Normalization Options

### Minimum TTL

When **Reset TTL** is greater than or equal to the value set for this option, specifies the following:

- the minimum value the system will permit in the IPv4 Time to Live (TTL) field when **Normalize IPv4** is enabled; a lower value results in normalizing the packet value for TTL to the value set for **Reset TTL**
- the minimum value the system will permit in the IPv6 Hop Limit field when **Normalize IPv6** is enabled; a lower value results in normalizing the packet value for Hop Limit to the value set for **Reset TTL**

The system assumes a value of 1 when the field is empty.



**Note** For Firewall Threat Defense routed and transparent interfaces, the **Minimum TTL** and **Reset TTL** options are ignored. The maximum TTL for a connection is 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 prevents TTL evasion attacks.

When the packet decoding **Detect Protocol Header Anomalies** option is enabled, you can enable the following rules in the decoder rule category to generate events and, in an inline deployment, drop offending packets for this option:

- You can enable rule 116:428 to trigger when the system detects an IPv4 packet with a TTL less than the specified minimum.
- You can enable rule 116:270 to trigger when the system detects an IPv6 packet with a hop limit that is less than the specified minimum.

### Reset TTL

When set to a value greater than or equal to **Minimum TTL**, normalizes the following:

- the IPv4 TTL field when **Normalize IPv4** is enabled
- the IPv6 Hop Limit field when **Normalize IPv6** is enabled

The system normalizes the packet by changing its TTL or Hop Limit value to the value set for this option when the packet value is less than **Minimum TTL**. Leaving this field blank, or setting it to 0, or to any value less than **Minimum TTL**, disables the option.

### Normalize IPv4

Enables normalization of IPv4 traffic. The system also normalizes the TTL field as needed when:

- this option is enabled, and
- the value set for **Reset TTL** enables TTL normalization.

You can also enable additional IPv4 options when this option is enabled.

When you enable this option, the system performs the following base IPv4 normalizations:

- truncates packets with excess payload to the datagram length specified in the IP header
- clears the Differentiated Services (DS) field, formerly known as the Type of Service (TOS) field
- sets all option octets to 1 (No Operation)

This option is ignored for Firewall Threat Defense routed and transparent interfaces. Firewall Threat Defense devices will drop any RSVP packet that contains IP options other than the router alert, end of options list (EOOL), and no operation (NOP) options on any routed or transparent interface.

### Normalize Don't Fragment Bit

Clears the single-bit Don't Fragment subfield of the IPv4 Flags header field. Enabling this option allows a downstream router to fragment packets if necessary instead of dropping them; enabling this option can also

prevent evasions based on crafting packets to be dropped. You must enable **Normalize IPv4** to select this option.

#### **Normalize Reserved Bit**

Clears the single-bit Reserved subfield of the IPv4 Flags header field. You would typically enable this option. You must enable **Normalize IPv4** to select this option.

#### **Normalize TOS Bit**

Clears the one byte Differentiated Services field, formerly known as Type of Service. You must enable **Normalize IPv4** to select this option.

#### **Normalize Excess Payload**

Truncates packets with excess payload to the datagram length specified in the IP header plus the Layer 2 (for example, Ethernet) header, but does not truncate below the minimum frame length. You must enable **Normalize IPv4** to select this option.

This option is ignored for Firewall Threat Defense routed and transparent interfaces. Packets with excess payload are always dropped on these interfaces.

#### **Normalize IPv6**

Sets all Option Type fields in the Hop-by-Hop Options and Destination Options extension headers to 00 (Skip and continue processing). The system also normalizes the Hop Limit field as needed when this option is enabled and the value set for **Reset TTL** enables hop limit normalization.

#### **Normalize ICMPv4**

Clears the 8-bit Code field in Echo (Request) and Echo Reply messages in ICMPv4 traffic.

#### **Normalize ICMPv6**

Clears the 8-bit Code field in Echo (Request) and Echo Reply messages in ICMPv6 traffic.

#### **Normalize/Clear Reserved Bits**

Clears the Reserved bits in the TCP header.

#### **Normalize/Clear Option Padding Bytes**

Clears any TCP option padding bytes.

#### **Clear Urgent Pointer if URG=0**

Clears the 16-bit TCP header Urgent Pointer field if the urgent (URG) control bit is not set.

#### **Clear Urgent Pointer/URG on Empty Payload**

Clears the TCP header Urgent Pointer field and the URG control bit if there is no payload.

#### **Clear URG if Urgent Pointer is Not Set**

Clears the TCP header URG control bit if the urgent pointer is not set.

### Normalize Urgent Pointer

Sets the two-byte TCP header Urgent Pointer field to the payload length if the pointer is greater than the payload length.

### Normalize TCP Payload

Enables normalization of the TCP Data field to ensure consistency in retransmitted data. Any segment that cannot be properly reassembled is dropped.

### Remove Data on SYN

Removes data in synchronization (SYN) packets if your TCP operating system policy is **not** Mac OS.

This option also disables rule 129:2, which can otherwise trigger when the TCP stream preprocessor **Policy** option is not set to **Mac OS**.

### Remove Data on RST

Removes any data from a TCP reset (RST) packet.

### Trim Data to Window

Trims the TCP Data field to the size specified in the Window field.

### Trim Data to MSS

Trims the TCP Data field to the Maximum Segment Size (MSS) if the payload is longer than MSS.

### Block Unresolvable TCP Header Anomalies

When you enable this option, the system blocks anomalous TCP packets that, if normalized, would be invalid and likely would be blocked by the receiving host. For example, the system blocks any SYN packet transmitted subsequent to an established session.

The system also drops any packet that matches any of the following TCP stream preprocessor rules, regardless of whether the rules are enabled:

- 129:1
- 129:3
- 129:4
- 129:6
- 129:8
- 129:11
- 129:14 through 129:19

The Total Blocked Packets performance graph tracks the number of packets blocked in inline deployments and, in passive deployments and inline deployments in tap mode, the number that would have been blocked in an inline deployment.

### Explicit Congestion Notification

Enables per-packet or per-stream normalization of Explicit Congestion Notification (ECN) flags as follows:

- select **Packet** to clear ECN flags on a per-packet basis regardless of negotiation
- select **Stream** to clear ECN flags on a per-stream basis if ECN use was not negotiated

If you select **Stream**, you must also ensure that the TCP stream preprocessor **Require TCP 3-Way Handshake** option is enabled for this normalization to take place.

### Clear Existing TCP Options

Enables **Allow These TCP Options**.

### Allow These TCP Options

Disables normalization of specific TCP options you allow in traffic.

The system does not normalize options that you explicitly allow. It normalizes options that you do not explicitly allow by setting the options to No Operation (TCP Option 1).

The system always allows the following options regardless of the configuration of **Allow These TCP Options** because they are commonly used for optimal TCP performance:

- Maximum Segment Size (MSS)
- Window Scale
- Time Stamp TCP

The system does not automatically allow other less commonly used options.

You can allow specific options by configuring a comma-separated list of option keywords, option numbers, or both as shown in the following example:

```
sack, echo, 19
```

Specifying an option keyword is the same as specifying the number for one or more TCP options associated with the keyword. For example, specifying `sack` is the same as specifying TCP options 4 (Selective Acknowledgment Permitted) and 5 (Selective Acknowledgment). Option keywords are not case sensitive.

You can also specify `any`, which allows all TCP options and effectively disables normalization of all TCP options.

The following table summarizes how you can specify TCP options to allow. If you leave the field empty, the system allows only the MSS, Window Scale, and Time Stamp options.

Specify...	To allow...
sack	TCP options 4 (Selective Acknowledgment Permitted) and 5 (Selective Acknowledgment)
echo	TCP options 6 (Echo Request) and 7 (Echo Reply)
partial_order	TCP options 9 (Partial Order Connection Permitted) and 10 (Partial Order Service Profile)

Specify...	To allow...
conn_count	TCP Connection Count options 11 (CC), 12 (CC.New), and 13 (CC.Echo)
alt_checksum	TCP options 14 (Alternate Checksum Request) and 15 (Alternate Checksum)
md5	TCP option 19 (MD5 Signature)
the option number, 2 to 255	a specific option, including options for which there is no keyword
any	all TCP options; this setting effectively disables TCP option normalization

When you do not specify `any` for this option, normalizations include the following:

- except MSS, Window Scale, Time Stamp, and any explicitly allowed options, sets all option bytes to No Operation (TCP Option 1)
- sets the Time Stamp octets to No Operation if Time Stamp is present but invalid, or valid but not negotiated
- blocks the packet if Time Stamp is negotiated but not present
- clears the Time Stamp Echo Reply (TSecr) option field if the Acknowledgment (ACK) control bit is not set
- sets the MSS and Window Scale options to No Operation (TCP Option 1) if the SYN control bit is not set

#### Related Topics

[Preprocessor Traffic Modification in Inline Deployments](#)  
[About Adaptive Profiles](#)

## Configuring Inline Normalization




---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

#### Before you begin

- If you want to normalize or drop offending packets, enable **Inline Mode** as described in [Preprocessor Traffic Modification in Inline Deployments](#). The managed device must also be deployed inline.

## Procedure

**Step 1** Choose **Policies > Security policies > Access Control**, and then click **Network Analysis Policy** or **Policies > Security policies > Intrusion**, and then click **Network Analysis Policies**.

### Note

If your custom user role limits access to the first path listed here, use the second path to access the policy.

**Step 2** Click **Snort 2 Version** next to the policy you want to edit.

**Step 3** Click **Edit** (✎) next to the policy you want to edit.

If **View** (👁) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

**Step 4** Click **Settings** in the navigation panel (NOT the caret; click the word).

**Step 5** If **Inline Normalization** under **Transport/Network Layer Preprocessors** is disabled, click **Enabled**.

**Step 6** Click **Edit** (✎) next to **Inline Normalization**.

**Step 7** Set the options described in [The Inline Normalization Preprocessor, on page 7](#).

**Step 8** To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

## What to do next

- If you want the inline normalization Minimum TTL option to generate intrusion events, enable either or both packet decoder rules 116:429 (IPv4) and 116:270 (IPv6). For more information, see [Setting Intrusion Rule States](#), and [Inline Normalization Options, on page 7](#).
- Deploy configuration changes.

## Related Topics

[Layer Management](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

[Preprocessor Traffic Modification in Inline Deployments](#)

[About Adaptive Profiles](#)

# The IP Defragmentation Preprocessor



**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

When an IP datagram is broken into two or more smaller IP datagrams because it is larger than the maximum transmission unit (MTU), it is *fragmented*. A single IP datagram fragment may not contain enough information to identify a hidden attack. Attackers may attempt to evade detection by transmitting attack data in fragmented packets. The IP defragmentation preprocessor reassembles fragmented IP datagrams before the rules engine executes rules against them so the rules can more appropriately identify attacks in those packets. If fragmented datagrams cannot be reassembled, rules do not execute against them.

## IP Fragmentation Exploits

Enabling IP defragmentation helps you detect attacks against hosts on your network, like the teardrop attack, and resource consumption attacks against the system itself, like the Jolt2 attack.

The Teardrop attack exploits a bug in certain operating systems that causes them to crash when trying to reassemble overlapping IP fragments. When enabled and configured to do so, the IP defragmentation preprocessor identifies the overlapping fragments. The IP defragmentation preprocessor detects the first packets in an overlapping fragment attack such as Teardrop, but does not detect subsequent packets for the same attack.

The Jolt2 attack sends a large number of copies of the same fragmented IP packet in an attempt to overuse IP defragmentors and cause a denial of service attack. A memory usage cap disrupts this and similar attacks in the IP defragmentation preprocessor, and places the system self-preservation above exhaustive inspection. The system is not overwhelmed by the attack, remains operational, and continues to inspect network traffic.

Different operating systems reassemble fragmented packets in different ways. Attackers who can determine which operating systems your hosts are running can also fragment malicious packets so that a target host reassembles them in a specific manner. Because the system does not know which operating systems the hosts on your monitored network are running, the preprocessor may reassemble and inspect the packets incorrectly, thus allowing an exploit to pass through undetected. To mitigate this kind of attack, you can configure the defragmentation preprocessor to use the appropriate method of defragmenting packets for each host on your network.

Note that you can also use adaptive profile updates in a passive deployment to dynamically select target-based policies for the IP defragmentation preprocessor using host operating system information for the target host in a packet.

## Target-Based Defragmentation Policies

A host's operating system uses three criteria to determine which packet fragments to favor when reassembling the packet:

- the order in which the fragment was received by the operating system
- its offset (the fragment's distance, in bytes, from the beginning of the packet)
- its beginning and ending position compared to overlap fragments.

Although every operating system uses these criteria, different operating systems favor different fragments when reassembling fragmented packets. Therefore, two hosts with different operating systems on your network could reassemble the same overlapping fragments in entirely different ways.

An attacker, aware of the operating system of one of your hosts, could attempt to evade detection and exploit that host by sending malicious content hidden in overlapping packet fragments. This packet, when reassembled and inspected, seems innocuous, but when reassembled by the target host, contains a malicious exploit. However, if you configure the IP defragmentation preprocessor to be aware of the operating systems running

on your monitored network segment, it will reassemble the fragments the same way that the target host does, allowing it to identify the attack.

## IP Defragmentation Options

You can choose to simply enable or disable IP defragmentation; however, Cisco recommends that you specify the behavior of the enabled IP defragmentation preprocessor at a more granular level.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

You can configure the following global option:

### Preallocated Fragments

The maximum number of individual fragments that the preprocessor can process at once. Specifying the number of fragment nodes to preallocate enables static memory allocation.



---

**Caution** Processing an individual fragment uses approximately 1550 bytes of memory. If the preprocessor requires more memory to process the individual fragments than the predetermined allowable memory limit for the managed device, the memory limit for the device takes precedence.

---

You can configure the following options for each IP defragmentation policy:

### Networks

The IP address of the host or hosts to which you want to apply the defragmentation policy.

You can specify a single IP address or address block, or a comma-separated list of either or both. You can specify up to 255 total profiles, including the default policy.

Note that the `default` setting in the default policy specifies all IP addresses on your monitored network segment that are not covered by another target-based policy. Therefore, you cannot and do not need to specify an IP address or CIDR block/prefix length for the default policy, and you cannot leave this setting blank in another policy or use address notation to represent `any` (for example, `0.0.0.0/0` or `::/0`).

### Policy

The defragmentation policy you want to use for a set of hosts on your monitored network segment.

You can select one of seven defragmentation policies, depending on the operating system of the target host. The following table lists the seven policies and the operating systems that use each one. The First and Last policy names reflect whether those policies favor original or subsequent overlapping packets.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

**Table 1: Target-Based Defragmentation Policies**

Policy	Operating Systems
BSD	AIX FreeBSD IRIX VAX/VMS
BSD-right	HP JetDirect
First	Mac OS HP-UX
Linux	Linux OpenBSD
Last	Cisco IOS
Solaris	SunOS
Windows	Windows

**Timeout**

Specifies the maximum amount of time, in seconds, that the preprocessor engine can use when reassembling a fragmented packet. If the packet cannot be reassembled within the specified time period, the preprocessor engine stops attempting to reassemble the packet and discards received fragments.

**Min TTL**

Specifies the minimum acceptable TTL value a packet may have. This option detects TTL-based insertion attacks.

You can enable rule 123:11 to generate events and, in an inline deployment, drop offending packets for this option.

**Detect Anomalies**

Identifies fragmentation problems such as overlapping fragments.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

You can enable the following rules to generate events and, in an inline deployment, drop offending packets for this option:

- 123:1 through 123:4
- 123:5 (BSD policy)
- 123:6 through 123:8

### Overlap Limit

Specifies that when the configured number of overlapping segments in a session has been detected, defragmentation stops for that session.

You must enable **Detect Anomalies** to configure this option. A blank value disables this option. A value of 0 specifies an unlimited number overlapping segments.

This option is ignored for Firewall Threat Defense routed and transparent interfaces. Overlapping fragments are always dropped on those interfaces.

You can enable rule 123:12 to generate events and, in an inline deployment, drop offending packets for this option.

### Minimum Fragment Size

Specifies that when a non-last fragment smaller than the configured number of bytes has been detected, the packet is considered malicious.

You must enable **Detect Anomalies** to configure this option. A blank value disables this option. A value of 0 specifies an unlimited number of bytes.

You can enable rule 123:13 to generate events and, in an inline deployment, drop offending packets for this option.

## Configuring IP Defragmentation



---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

### Before you begin

- Confirm that any networks you want to identify in a custom target-based policy match or are a subset of the networks, zones, and VLANs handled by its parent network analysis policy. See [Advanced Settings for Network Analysis Policies](#) for more information.

### Procedure

---

**Step 1** Choose **Policies > Security policies > Access Control**, and then click **Network Analysis Policy** or **Policies > Security policies > Intrusion**, and then click **Network Analysis Policies**.

**Note**

If your custom user role limits access to the first path listed here, use the second path to access the policy.

**Step 2** Click **Snort 2 Version** next to the policy you want to edit.

**Step 3** Click **Edit** (✎) next to the policy you want to edit.

If **View** (👁) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

- Step 4** Click **Settings** in the navigation panel.
- Step 5** If **IP Defragmentation** under **Transport/Network Layer Preprocessors** is disabled, click **Enabled**.
- Step 6** Click **Edit** (✎) next to **IP Defragmentation**.
- Step 7** Optionally, enter a value in the **Preallocated Fragments** field.
- Step 8** You have the following choices:
- Add a server profile — Click **Add** (+) next to **Servers** on the left side of the page, then enter a value in the **Host Address** field and click **OK**. You can specify a single IP address or address block, or a comma-separated list of either or both. You can create a total of 255 target-based policies including the default policy.
  - Edit a server profile — Click the configured address for under **Servers** on the left side of the page, or click **default**.
  - Delete a profile — Click **Delete** (🗑) next to the policy.
- Step 9** Modify the options described in [IP Defragmentation Options, on page 15](#).
- Step 10** To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.
- If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

---

### What to do next

- If you want to generate events and, in an inline deployment, drop offending packets, enable IP defragmentation rules (GID 123). For more information, see [Setting Intrusion Rule States and IP Defragmentation Options, on page 15](#).
- Deploy configuration changes.

### Related Topics

[Layer Basics](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

## The Packet Decoder




---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

Before sending captured packets to a preprocessor, the system first sends the packets to the packet decoder. The packet decoder converts packet headers and payloads into a format that preprocessors and the rules engine can easily use. Each stack layer is decoded in turn, beginning with the data link layer and continuing through the network and transport layers.

## Packet Decoder Options

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

### Decode GTP Data Channel

Decodes the encapsulated GTP (General Packet Radio Service [GPRS] Tunneling Protocol) data channel. By default, the decoder decodes version 0 data on port 3386 and version 1 data on port 2152. You can use the `GTP_PORTS` default variable to modify the ports that identify encapsulated GTP traffic.

You can enable rules 116:297 and 116:298 to generate events and, in an inline deployment, drop offending packets for this option.

### Detect Teredo on Non-Standard Ports

Inspects Teredo tunneling of IPv6 traffic that is identified on a UDP port other than port 3544.

The system always inspects IPv6 traffic when it is present. By default, IPv6 inspection includes the 4in6, 6in4, 6to4, and 6in6 tunneling schemes, and also includes Teredo tunneling when the UDP header specifies port 3544.

In an IPv4 network, IPv4 hosts can use the Teredo protocol to tunnel IPv6 traffic through an IPv4 Network Address Translation (NAT) device. Teredo encapsulates IPv6 packets within IPv4 UDP datagrams to permit IPv6 connectivity behind an IPv4 NAT device. The system normally uses UDP port 3544 to identify Teredo traffic. However, an attacker could use a non-standard port in an attempt to avoid detection. You can enable **Detect Teredo on Non-Standard Ports** to cause the system to inspect all UDP payloads for Teredo tunneling.

Teredo decoding occurs only on the first UDP header, and only when IPv4 is used for the outer network layer. When a second UDP layer is present after the Teredo IPv6 layer because of UDP data encapsulated in the IPv6 data, the rules engine uses UDP intrusion rules to analyze both the inner and outer UDP layers.

Note that intrusion rules 12065, 12066, 12067, and 12068 in the **policy-other** rule category detect, but do not decode, Teredo traffic. Optionally, you can use these rules to drop Teredo traffic in an inline deployment; however, you should ensure that these rules are disabled or set to generate events without dropping traffic when you enable **Detect Teredo on Non-Standard Ports**.

### Detect Excessive Length Value

Detects when the packet header specifies a packet length that is greater than the actual packet length.

This option is ignored for Firewall Threat Defense routed, transparent, and inline interfaces. Packets that have excessive header length are always dropped. However, this option does apply to Firewall Threat Defense inline tap and passive interfaces.

You can enable rules 116:6, 116:47, 116:97, and 116:275 to generate events and, in an inline deployment, drop offending packets for this option.

### Detect Invalid IP Options

Detects invalid IP header options to identify exploits that use invalid IP options. For example, there is a denial of service attack against a firewall which causes the system to freeze. The firewall attempts to parse invalid Timestamp and Security IP options and fails to check for a zero length, which causes an irrecoverable infinite loop. The rules engine identifies the zero length option, and provides information you can use to mitigate the attack at the firewall.

Firewall Threat Defense devices will drop any RSVP packet that contains IP options other than the router alert, end of options list (EOOL), and no operation (NOP) options on any routed or transparent interface. For inline, inline tap, or passive interfaces, IP options will be handled as described above.

You can enable rules 116:4 and 116:5 to generate events and, in an inline deployment, drop offending packets for this option.

### Detect Experimental TCP Options

Detects TCP headers with experimental TCP options. The following table describes these options.

TCP Option	Description
9	Partial Order Connection Permitted
10	Partial Order Service Profile
14	Alternate Checksum Request
15	Alternate Checksum Data
18	Trailer Checksum
20	Space Communications Protocol Standards (SCPS)
21	Selective Negative Acknowledgements (SCPS)
22	Record Boundaries (SCPS)
23	Corruption (SPCS)
24	SNAP
26	TCP Compression Filter

Because these are experimental options, some systems do not account for them and may be open to exploits.



**Note** In addition to the experimental options listed in the above table, the system considers any TCP option with an option number greater than 26 to be experimental.

You can enable rule 116:58 to generate events and, in an inline deployment, drop offending packets for this option.

### Detect Obsolete TCP Options

Detects TCP headers with obsolete TCP options. Because these are obsolete options, some systems do not account for them and may be open to exploits. The following table describes these options.

TCP Option	Description
6	Echo
7	Echo Reply

TCP Option	Description
16	Skeeter
17	Bubba
19	MD5 Signature
25	Unassigned

You can enable rule 116:57 to generate events and, in an inline deployment, drop offending packets for this option.

**Detect T/TCP**

Detects TCP headers with the CC.ECHO option. The CC.ECHO option confirms that TCP for Transactions (T/TCP) is being used. Because T/TCP header options are not in widespread use, some systems do not account for them and may be open to exploits.

You can enable rule 116:56 to generate events and, in an inline deployment, drop offending packets for this option.

**Detect Other TCP Options**

Detects TCP headers with invalid TCP options not detected by other TCP decoding event options. For example, this option detects TCP options with the incorrect length or with a length that places the option data outside the TCP header.

This option is ignored for Firewall Threat Defense routed and transparent interfaces. Packets that have invalid TCP options are always dropped.

You can enable rules 116:54, 116:55, and 116:59 to generate events and, in an inline deployment, drop offending packets for this option.

**Detect Protocol Header Anomalies**

Detects other decoding errors not detected by the more specific IP and TCP decoder options. For example, the decoder might detect a malformed data-link protocol header.

This option is ignored for Firewall Threat Defense routed, transparent, and inline interfaces. Packets that have header anomalies are always dropped. However, this option does apply to Threat Defense inline tap and passive interfaces.

To generate events and, in an inline deployment, drop offending packets for this option, you can enable any of the following rules:

GID:SID	Generates an event if:
116:467	The packet is smaller than the minimum size of a packet encapsulated with a Cisco FabricPath header.
116:468	The Cisco Meta Data (CMD) field in the header contains a header length smaller than the minimum size of a valid CMD header. The CMD field is associated with the Cisco Trustsec protocol.

GID:SID	Generates an event if:
116:469	The CMD field in the header contains an invalid field length.
116:470	The CMD field in the header contains an invalid Security Group Tag (SGT) option type.
116:471	The CMD field in the header contains an SGT with a reserved value.

You can also enable any packet decoder rule not associated with other packet decoder options.

#### Related Topics

[Predefined Default Variables](#)

## Configuring Packet Decoding



**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

### Procedure

**Step 1** Choose **Policies > Security policies > Access Control**, and then click **Network Analysis Policy** or **Policies > Security policies > Intrusion**, and then click **Network Analysis Policies**.

**Note**

If your custom user role limits access to the first path listed here, use the second path to access the policy.

**Step 2** Click **Snort 2 Version** next to the policy you want to edit.

**Step 3** Click **Edit** (✎) next to the policy you want to edit.

If **View** (👁) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

**Step 4** Click **Settings** in the navigation panel.

**Step 5** If **Packet Decoding** under **Transport/Network Layer Preprocessors** is disabled, click **Enabled**.

**Step 6** Click **Edit** (✎) next to **Packet Decoding**.

**Step 7** Enable or disable the options described in [Packet Decoder Options, on page 19](#).

**Step 8** To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

### What to do next

- If you want to generate events and, in an inline deployment, drop offending packets, enable packet decoder rules (GID 116). For more information, see [Setting Intrusion Rule States](#) and [Packet Decoder Options, on page 19](#).
- Deploy configuration changes.

### Related Topics

[Layer Basics](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

## TCP Stream Preprocessing



---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

The TCP protocol defines various states in which connections can exist. Each TCP connection is identified by the source and destination IP addresses and source and destination ports. TCP permits only one connection with the same connection parameter values to exist at a time.

## State-Related TCP Exploits

If you add the `flow` keyword with the `established` argument to an intrusion rule, the intrusion rules engine inspects packets matching the rule and the flow directive in stateful mode. Stateful mode evaluates only the traffic that is part of a TCP session established with a legitimate three-way handshake between a client and server.

You can configure the system so that the preprocessor detects any TCP traffic that cannot be identified as part of an established TCP session, although this is not recommended for typical use because the events would quickly overload the system and not provide meaningful data.

Attacks like `stick` and `snot` use the system's extensive rule sets and packet inspection against itself. These tools generate packets based on the patterns in Snort-based intrusion rules, and send them across the network. If your rules do not include the `flow` or `flowbits` keyword to configure them for stateful inspection, each packet will trigger the rule, overwhelming the system. Stateful inspection allows you to ignore these packets because they are not part of an established TCP session and do not provide meaningful information. When performing stateful inspection, the rules engine detects only those attacks that are part of an established TCP session, allowing analysts to focus on these rather than the volume of events caused by `stick` or `snot`.

## Target-Based TCP Policies

Different operating systems implement TCP in different ways. For example, Windows and some other operating systems require a TCP reset segment to have a precise TCP sequence number to reset a session, while Linux and other operating systems permit a range of sequence numbers. In this example, the stream preprocessor must understand exactly how the destination host will respond to the reset based on the sequence number. The stream preprocessor stops tracking the session only when the destination host considers the reset to be valid, so an attack cannot evade detection by sending packets after the preprocessor stops inspecting the

stream. Other variations in TCP implementations include such things as whether an operating system employs a TCP timestamp option and, if so, how it handles the timestamp, and whether an operating system accepts or ignores data in a SYN packet.

Different operating systems also reassemble overlapping TCP segments in different ways. Overlapping TCP segments could reflect normal retransmissions of unacknowledged TCP traffic. They could also represent an attempt by an attacker, aware of the operating system of one of your hosts, to evade detection and exploit that host by sending malicious content hidden in overlapping segments. However, you can configure the stream preprocessor to be aware of the operating systems running on your monitored network segment so it reassembles segments the same way the target host does, allowing it to identify the attack.

You can create one or more TCP policies to tailor TCP stream inspection and reassembly to the different operating systems on your monitored network segment. For each policy, you identify one of thirteen operating system policies. You bind each TCP policy to a specific IP address or address block using as many TCP policies as you need to identify any or all of the hosts using a different operating system. The default TCP policy applies to any hosts on the monitored network that you do not identify in any other TCP policy, so there is no need to specify an IP address or address block for the default TCP policy.

Note that you can also use adaptive profile updates in a passive deployment to dynamically select target-based policies for the TCP stream preprocessor using host operating system information for the target host in a packet.

## TCP Stream Reassembly

The stream preprocessor collects and reassembles all the packets that are part of a TCP session's server-to-client communication stream, client-to-server communication stream, or both. This allows the rules engine to inspect the stream as a single, reassembled entity rather than inspecting only the individual packets that are part of a given stream.

Stream reassembly allows the rules engine to identify stream-based attacks, which it may not detect when inspecting individual packets. You can specify which communication streams the rules engine reassembles based on your network needs. For example, when monitoring traffic on your web servers, you may only want to inspect client traffic because you are much less likely to receive malicious traffic from your own web server.

In each TCP policy, you can specify a comma-separated list of ports to identify the traffic for the stream preprocessor to reassemble. If adaptive profile updates are enabled, you can also list services that identify traffic to reassemble, either as an alternative to ports or in combination with ports.

You can specify ports, services, or both. You can specify separate lists of ports for any combination of client ports, server ports, and both. You can also specify separate lists of services for any combination of client services, server services, and both. For example, assume that you wanted to reassemble the following:

- SMTP (port 25) traffic from the client
- FTP server responses (port 21)
- telnet (port 23) traffic in both directions

You could configure the following:

- For client ports, specify 23, 25
- For server ports, specify 21, 23

Or, instead, you could configure the following:

- For client ports, specify `25`
- For server ports, specify `21`
- For both ports, specify `23`

Additionally, consider the following example which combines ports and services and would be valid when adaptive profile updates are enabled:

- For client ports, specify `23`
- For client services, specify `smtp`
- For server ports, specify `21`
- For server services, specify `telnet`

Negating a port (for example, `!80`) can improve performance by preventing the TCP stream processor from processing traffic for that port.

Although you can also specify `all` as the argument to provide reassembly for all ports, Cisco does **not** recommend setting ports to `all` because it may increase the amount of traffic inspected by this processor and slow performance unnecessarily.

TCP reassembly automatically and transparently includes ports that you add to other preprocessors. However, if you do explicitly add ports to TCP reassembly lists that you have added to other preprocessor configurations, these additional ports are handled normally. This includes port lists for the following preprocessors:

- FTP/Telnet (server-level FTP)
- DCE/RPC
- HTTP Inspect
- SMTP
- Session Initiation Protocol
- POP
- IMAP
- SSL

Note that reassembling additional traffic types (client, server, both) increases resource demands.

## TCP Stream Preprocessing Options

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

You can configure the following global TCP option:

### Packet Type Performance Boost

Enables ignoring TCP traffic for all ports and application protocols that are not specified in enabled intrusion rules, except when a TCP rule with both the source and destination ports set to `any` has a `flow` or `flowbits` option. This performance improvement could result in missed attacks.

You can configure the following options for each TCP policy.

### Network

Specifies the host IP addresses to which you want to apply the TCP stream reassembly policy.

You can specify a single IP address or address block. You can specify up to 255 total profiles including the default policy.

Note that the `default` setting in the default policy specifies all IP addresses on your monitored network segment that are not covered by another target-based policy. Therefore, you cannot and do not need to specify an IP address or CIDR block/prefix length for the default policy, and you cannot leave this setting blank in another policy or use address notation to represent `any` (for example, `0.0.0.0/0` or `::/0`).

### Policy

Identifies the TCP policy operating system of the target host or hosts. If you select a policy other than **Mac OS**, the system removes the data from the synchronization (SYN) packets and disables event generation for rule 129:2. Note that enabling the inline normalization preprocessor **Remove Data on SYN** option also disables rule 129:2.

The following table identifies the operating system policies and the host operating systems that use each.

**Table 2: TCP Operating System Policies**

Policy	Operating Systems
First	unknown OS
Last	Cisco IOS
BSD	AIX FreeBSD OpenBSD
Linux	Linux 2.4 kernel Linux 2.6 kernel
Old Linux	Linux 2.2 and earlier kernel
Windows	Windows 98 Windows NT Windows 2000 Windows XP
Windows 2003	Windows 2003
Windows Vista	Windows Vista
Solaris	Solaris OS SunOS

Policy	Operating Systems
IRIX	SGI Irix
HPUX	HP-UX 11.0 and later
HPUX 10	HP-UX 10.2 and earlier
Mac OS	Mac OS 10 (Mac OS X)



**Tip** The First operating system policy could offer some protection when you do not know the host operating system. However, it may result in missed attacks. You should edit the policy to specify the correct operating system if you know it.

#### Timeout

The number of seconds between 1 and 86400 the intrusion rules engine keeps an inactive stream in the state table. If the stream is not reassembled in the specified time, the intrusion rules engine deletes it from the state table.



**Note** If your managed device is deployed on a segment where the network traffic is likely to reach the device's bandwidth limits, you should consider setting this value higher (for example, to 600 seconds) to lower the amount of processing overhead.

Firewall Threat Defense devices ignore this option and, instead, use the settings in the advanced access control **Threat Defense Service Policy**. See [Configure a Service Policy Rule](#) for more information.

#### Maximum TCP Window

Specifies the maximum TCP window size between 1 and 1073725440 bytes allowed as specified by a receiving host. Setting the value to 0 disables checking for the TCP window size.



**Caution** The upper limit is the maximum window size permitted by RFC, and is intended to prevent an attacker from evading detection, but setting a significantly large maximum window size could result in a self-imposed denial of service.

When **Stateful Inspection Anomalies** is enabled, you can enable rule 129:6 to generate events and, in an inline deployment, drop offending packets for this option.

#### Overlap Limit

Specifies that when the configured number between 0 (unlimited) and 255 of overlapping segments in a session has been detected, segment reassembly stops for that session and, if **Stateful Inspection Anomalies** is enabled and the accompanying preprocessor rule is enabled, an event is generated.

You can enable rule 129:7 to generate events and, in an inline deployment, drop offending packets for this option.

### Flush Factor

In an inline deployment, specifies that when a segment of decreased size has been detected subsequent to the configured number between 1 and 2048 of segments of non-decreasing size, the system flushes segment data accumulated for detection. Setting the value to 0 disables detection of this segment pattern, which can indicate the end of a request or response. Note that the Inline Normalization **Normalize TCP Payload** option must be enabled for this option to be effective.

### Stateful Inspection Anomalies

Detects anomalous behavior in the TCP stack. When accompanying preprocessor rules are enabled, this may generate many events if TCP/IP stacks are poorly written.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

You can enable the following rules to generate events and, in an inline deployment, drop offending packets for this option:

- 129:1 through 129:5
- 129:6 (Mac OS only)
- 129:8 through 129:11
- 129:13 through 129:19

Note the following:

- for rule 129:6 to trigger you must also configure a value greater than 0 for **Maximum TCP Window**.
- for rules 129:9 and 129:10 to trigger you must also enable **TCP Session Hijacking**.

### TCP Session Hijacking

Detects TCP session hijacking by validating the hardware (MAC) addresses detected from both sides of a TCP connection during the 3-way handshake against subsequent packets received on the session. When the MAC address for one side or the other does not match, if **Stateful Inspection Anomalies** is enabled and one of the two corresponding preprocessor rules are enabled, the system generates events.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

You can enable rules 129:9 and 129:10 to generate events and, in an inline deployment, drop offending packets for this option. Note that for either of these rules to generate events you must also enable **Stateful Inspection Anomalies**.

### Consecutive Small Segments

When **Stateful Inspection Anomalies** is enabled, specifies a maximum number of 1 to 2048 consecutive small TCP segments allowed. Setting the value to 0 disables checking for consecutive small segments.

You must set this option together with the **Small Segment Size** option, either disabling both or setting a non-zero value for both. Note that receiving as many as 2000 consecutive segments, even if each segment was 1 byte in length, without an intervening ACK would be far more consecutive segments than you would normally expect.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

You can enable rule 129:12 to generate events and, in an inline deployment, drop offending packets for this option.

### Small Segment Size

When **Stateful Inspection Anomalies** is enabled, specifies the 1 to 2048 byte TCP segment size that is considered small. Setting the value to 0 disables specifying the size of a small segment.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

You must set this option together with the **Consecutive Small Segments** option, either disabling both or setting a non-zero value for both. Note that a 2048 byte TCP segment is larger than a normal 1500 byte Ethernet frame.

### Ports Ignoring Small Segments

When **Stateful Inspection Anomalies**, **Consecutive Small Segments**, and **Small Segment Size** are enabled, specifies a comma-separated list of one or more ports that ignore small TCP segment detection. Leaving this option blank specifies that no ports are ignored.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

You can add any port to the list, but the list only affects ports specified in one of the **Perform Stream Reassembly on** port lists in the TCP policy.

### Require TCP 3-Way Handshake

Specifies that sessions are treated as established only upon completion of a TCP three-way handshake. Disable this option to increase performance, protect from SYN flood attacks, and permit operation in a partially asynchronous environment. Enable it to avoid attacks that attempt to generate false positives by sending information that is not part of an established TCP session.

You can enable rule 129:20 to generate events and, in an inline deployment, drop offending packets for this option.

### 3-Way Handshake Timeout

Specifies the number of seconds between 0 (unlimited) and 86400 (twenty-four hours) by which a handshake must be completed when **Require TCP 3-Way Handshake** is enabled. You must enable **Require TCP 3-Way Handshake** to modify the value for this option.

For Firepower Software devices and Firewall Threat Defense inline, inline tap, and passive interfaces, the default is 0. For Firewall Threat Defense routed and transparent interfaces, the timeout is always 30 seconds; the value configured here is ignored.

### Packet Size Performance Boost

Sets the preprocessor to not queue large packets in the reassembly buffer. This performance improvement could result in missed attacks. Disable this option to protect against evasion attempts using small packets of one to twenty bytes. Enable it when you are assured of no such attacks because all traffic is comprised of very large packets.

**Legacy Reassembly**

Sets the stream preprocessor to emulate the deprecated Stream 4 preprocessor when reassembling packets, which lets you compare events reassembled by the stream preprocessor to events based on the same data stream reassembled by the Stream 4 preprocessor.

**Asynchronous Network**

Specifies whether the monitored network is an asynchronous network, that is, a network where the system sees only half the traffic. When this option is enabled, the system does not reassemble TCP streams to increase performance.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

**Perform Stream Reassembly on Client Ports**

Enables stream reassembly based on ports for the client side of the connection. In other words, it reassembles streams destined for web servers, mail servers, or other IP addresses typically defined by the IP addresses specified in \$HOME\_NET. Use this option when you expect malicious traffic to originate from clients.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

**Perform Stream Reassembly on Client Services**

Enables stream reassembly based on services for the client side of the connection. Use this option when you expect malicious traffic to originate from clients.

At least one client detector must be enabled for each client service you select. By default, all Cisco-provided detectors are activated. If no detector is enabled for an associated client application, the system automatically enables all Cisco-provided detectors for the application; if none exist, the system enables the most recently modified user-defined detector for the application.

This feature requires Protection and Control licenses.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

**Perform Stream Reassembly on Server Ports**

Enables stream reassembly based on ports for the server side of the connection only. In other words, it reassembles streams originating from web servers, mail servers, or other IP addresses typically defined by the IP addresses specified in \$EXTERNAL\_NET. Use this option when you want to watch for server side attacks. You can disable this option by not specifying ports.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.




---

**Note** For a thorough inspection of a service, add the service name in the Perform Stream Reassembly on Server Services field in addition to adding the port number in the Perform Stream Reassembly on Server Ports field. For example, add 'HTTP' service in the Perform Stream Reassembly on Server Services field to inspect HTTP service in addition to adding port number 80 in the Perform Stream Reassembly on Server Ports field.

---

**Perform Stream Reassembly on Server Services**

Enables stream reassembly based on services for the server side of the connection only. Use this option when you want to watch for server side attacks. You can disable this option by not specifying services.

At least one detector must be enabled. By default, all Cisco-provided detectors are activated. If no detector is enabled for a service, the system automatically enables all Cisco-provided detectors for the associated application protocol; if none exist, the system enables the most recently modified user-defined detector for the application protocol.

This feature requires Protection and Control licenses.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

### Perform Stream Reassembly on Both Ports

Enables stream reassembly based on ports for both the client and server side of the connection. Use this option when you expect that malicious traffic for the same ports may travel in either direction between clients and servers. You can disable this option by not specifying ports.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

### Perform Stream Reassembly on Both Services

Enables stream reassembly based on services for both the client and server side of the connection. Use this option when you expect that malicious traffic for the same services may travel in either direction between clients and servers. You can disable this option by not specifying services.

At least one detector must be enabled. By default, all Cisco-provided detectors are activated. If no detector is enabled for an associated client application or application protocol, the system automatically enables all Cisco-provided detectors for the application or application protocol; if none exist, the system enables the most recently modified user-defined detector for the application or application protocol.

This feature requires Protection and Control licenses.

This option is ignored for Firewall Threat Defense routed and transparent interfaces.

### Troubleshooting Options: Maximum Queued Bytes

Support might ask you during a troubleshooting call to specify the amount of data that can be queued on one side of a TCP connection. A value of 0 specifies an unlimited number of bytes.



---

**Caution** Changing the setting for this troubleshooting option will affect performance and should be done only with Support guidance.

---

### Troubleshooting Options: Maximum Queued Segments

Support might ask you during a troubleshooting call to specify the maximum number of bytes of data segments that can be queued on one side of a TCP connection. A value of 0 specifies an unlimited number of data segment bytes.



---

**Caution** Changing the setting for this troubleshooting option will affect performance and should be done only with Support guidance.

---

### Related Topics

[Activating and Deactivating Detectors](#)

[Layer Management](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

## Configuring TCP Stream Preprocessing




---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

### Before you begin

- Confirm that networks you want to identify in a custom target-based policy match or are a subset of the networks, zones, and VLANs handled by its parent network analysis policy. See [Advanced Settings for Network Analysis Policies](#) for more information.

### Procedure

---

**Step 1** Choose **Policies > Security policies > Access Control**, and then click **Network Analysis Policy** or **Policies > Security policies > Intrusion**, and then click **Network Analysis Policies**.

**Note**

If your custom user role limits access to the first path listed here, use the second path to access the policy.

**Step 2** Click **Snort 2 Version** next to the policy you want to edit.

**Step 3** Click **Edit** (✎) next to the policy you want to modify.

If **View** (👁) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

**Step 4** Click **Settings** in the navigation panel on the left.

**Step 5** If the **TCP Stream Configuration** setting is disabled under **Transport/Network Layer Preprocessors**, enable it by clicking **Enabled**.

**Step 6** Click **Edit** (✎) next to **TCP Stream Configuration**.

**Step 7** Check or clear the **Packet Type Performance Boost** check box in the **Global Settings** section.

**Step 8** You can:

- Add a target-based policy — Click **Add** (+) next to **Hosts** in the Targets section. Specify one or more IP addresses in the **Host Address** field. You can specify a single IP address or address block. You can create a total of 255 target-based policies including the default policy. When done, click **OK**.
- Edit an exist target-based policy — Under **Hosts**, click on the address for the policy you want to edit, or click default to edit the **default** configuration values.
- Modify the TCP Stream Preprocessing options — See [TCP Stream Preprocessing Options, on page 25](#).

**Caution**

Do not modify **Maximum Queued Bytes** or **Maximum Queued Segments** unless instructed to do so by Support.

**Tip**

To modify stream reassembly settings based on client, server, or both services, click inside the field you want to modify or click **Edit** next to the field. Use arrow to move services between the **Available** and **Enabled** lists in the pop-up window, then click **OK**.

- Delete an existing target-based policy — Click **Delete** () next to the policy you want to remove.

**Step 9** To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

---

**What to do next**

- If you want to generate events and, in an inline deployment, drop offending packets, enable TCP Stream preprocessor rules (GID 129). For more information, see [Setting Intrusion Rule States](#) and [TCP Stream Preprocessing Options, on page 25](#).
- Deploy configuration changes.

**Related Topics**

[Layer Management](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

## UDP Stream Preprocessing



---

**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

---

UDP stream preprocessing occurs when the rules engine processes packets against a UDP rule that includes the `flow` keyword using any of the following arguments:

- `Established`
- `To Client`
- `From Client`
- `To Server`
- `From Server`

UDP data streams are not typically thought of in terms of *sessions*. UDP is a connectionless protocol that does not provide a means for two endpoints to establish a communication channel, exchange data, and close the channel. However, the stream preprocessor uses the source and destination IP address fields in the encapsulating IP datagram header and the port fields in the UDP header to determine the direction of flow and identify a

session. A session ends when a configurable timer is exceeded, or when either endpoint receives an ICMP message that the other endpoint is unreachable or the requested service is unavailable.

Note that the system does not generate events related to UDP stream preprocessing; however, you can enable related packet decoder rules to detect UDP protocol header anomalies.

#### Related Topics

[TCP Header Values and Stream Size](#)

## UDP Stream Preprocessing Options

### Timeout

Specifies the number of seconds the preprocessor keeps an inactive stream in the state table. If additional datagrams are not seen in the specified time, the preprocessor deletes the stream from the state table.

Firewall Threat Defense devices ignore this option and, instead, use the settings in the advanced access control **Threat Defense Service Policy**. See [Configure a Service Policy Rule](#) for more information.

### Packet Type Performance Boost

Sets to preprocessor to ignore UDP traffic for all ports and application protocols that are not specified in enabled rules, except when a UDP rule with both the source and destination ports set to `any` has a `flow` or `flowbits` option. This performance improvement could result in missed attacks.

#### Related Topics

[TCP Header Values and Stream Size](#)

## Configuring UDP Stream Preprocessing



**Note** This section applies to Snort 2 preprocessors. For information on Snort 3 inspectors, see <https://www.cisco.com/go/snort3-inspectors>.

### Procedure

**Step 1** Choose **Policies > Security policies > Access Control**, and then click **Network Analysis Policy** or **Policies > Security policies > Intrusion**, and then click **Network Analysis Policies**.

#### Note

If your custom user role limits access to the first path listed here, use the second path to access the policy.

**Step 2** Click **Snort 2 Version** next to the policy you want to edit.

**Step 3** Click **Edit** (✎) next to the policy you want to edit.

If **View** (👁) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

**Step 4** Click **Settings** in the navigation panel.

- Step 5** If **UDP Stream Configuration** under **Transport/Network Layer Preprocessors** is disabled, click **Enabled**.
- Step 6** Click **Edit** (✎) next to **UDP Stream Configuration**.
- Step 7** Set the options described in [UDP Stream Preprocessing Options, on page 34](#).
- Step 8** To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

---

### What to do next

- If you want to generate events and, in an inline deployment, drop offending packets, enable related packet decoder rules (GID 116). For more information, see [Setting Intrusion Rule States](#) and [The Packet Decoder, on page 18](#).
- Deploy configuration changes.

### Related Topics

[Layer Management](#)

[Conflicts and Changes: Network Analysis and Intrusion Policies](#)

[TCP Header Values and Stream Size](#)

