APPENDIX B

Signature Engines

This appendix describes the IPS signature engines. It contains the following sections:

Understanding Signature Engines

A signature engine is a component of the Cisco IPS that is designed to support many signatures in a certain category. An engine is composed of a parser and an inspector. Each engine has a set of parameters that have allowable ranges or sets of values.

Note The Cisco IPS engines support a standardized Regex.

Cisco IPS contains the following signature engines:

- AIC—Provides thorough analysis of web traffic. The AIC engine provides granular control over HTTP sessions to prevent abuse of the HTTP protocol. It allows administrative control over applications, such as instant messaging and gotomypc, that try to tunnel over specified ports. You can also use AIC to inspect FTP traffic and control the commands being issued. There are two AIC engines: AIC FTP and AIC HTTP.
- Atomic—The Atomic engines are combined into four engines with multi-level selections. You can combine Layer 3 and Layer 4 attributes within one signature, for example IP + TCP. The Atomic engine uses the standardized Regex support. The Atomic engines consist of the following types:
  - Atomic ARP—Inspects Layer 2 ARP protocol. The Atomic ARP engine is different because most engines are based on Layer 3 IP protocol.
  - Atomic IP Advanced—Inspects IPv6 Layer 3 and ICMPv6 Layer 4 traffic.
  - Atomic IP—Inspects IP protocol packets and associated Layer 4 transport protocols. This engine lets you specify values to match for fields in the IP and Layer 4 headers, and lets you use Regex to inspect Layer 4 payloads.

    Note All IP packets are inspected by the Atomic IP engine. This engine replaces the 4.x Atomic ICMP, Atomic IP Options, Atomic L3 IP, Atomic TCP, and Atomic UDP engines.
  - Atomic IPv6—Detects two IOS vulnerabilities that are stimulated by malformed IPv6 traffic.
- Fixed—Performs parallel regular expression matches up to a fixed depth, then stops inspection using a single regular expression table. There are three Fixed engines: ICMP, TCP, and UDP.
- Flood—Detects ICMP and UDP floods directed at hosts and networks. There are two Flood engines: Flood Host and Flood Net.
- Meta—Defines events that occur in a related manner within a sliding time interval. This engine processes events rather than packets.
- Multi String—Inspects Layer 4 transport protocols and payloads by matching several strings for one signature. This engine inspects stream-based TCP and single UDP and ICMP packets.
- Normalizer—Configures how the IP and TCP Normalizer functions and provides configuration for signature events related to the IP and TCP Normalizer. Allows you to enforce RFC compliance.
- Service—Deals with specific protocols. The Service engines are divided in to the following protocol types:
  - DNS—Inspects DNS (TCP and UDP) traffic.
  - FTP—Inspects FTP traffic.
  - FTP V2—Supports IOS IPS. This signature engine provides a protocol decode engine tuned for IOS IPS. If you try to use this engine, you receive an error message.
  - Generic—Decodes custom service and payload, and generically analyzes network protocols.
  - H225—Inspects VoIP traffic. Helps the network administrator make sure the SETUP message coming in to the VoIP network is valid and within the bounds that the policies describe. Is also helps make sure the addresses and Q.931 string fields such as url-ids, email-ids, and display information adhere to specific lengths and do not contain possible attack patterns.
  - HTTP—Inspects HTTP traffic. The WEBPORTS variable defines inspection port for HTTP traffic.
  - HTTP V2—Supports IOS IPS. This signature engine provides a protocol decode engine tuned for IOS IPS. If you try to use this engine, you receive an error message.

- IDENT—Inspects IDENT (client and server) traffic.
- MSRPC—Inspects MSRPC traffic.
- MSSQL—Inspects Microsoft SQL traffic.
- NTP—Inspects NTP traffic.
- P2P—Inspects P2P traffic.
- RPC—Inspects RPC traffic.
- SMB Advanced—Processes Microsoft SMB and Microsoft DCE/RPC (MSRPC) over SMB packets.

  Note The SMB engine has been replaced by the SMB Advanced engine. Even though the SMB engine is still visible in IDM, IME, and the CLI, its signatures have been obsoleted; that is, the new signatures have the obsoletes parameter set with the IDs of their corresponding old signatures. Use the new SMB Advanced engine to rewrite any custom signature that were in the SMB engine.

- SMPT V1—Supports IOS IPS.

  This signature engine provides a protocol decode engine tuned for IOS IPS. If you try to use this engine, you receive an error message.
- SNMP—Inspects SNMP traffic.
- SSH—Inspects SSH traffic.
- TNS—Inspects TNS traffic.

- State—Conducts stateful searches of strings in protocols such as SMTP. The state engine has a hidden configuration file that is used to define the state transitions so new state definitions can be delivered in a signature update.
- String—Searches on Regex strings based on ICMP, TCP, or UDP protocol. There are three String engines: String ICMP, String TCP, and String UDP.
- String XL—Searches on Regex strings based on ICMP, TCP, or UDP protocol.The String XL engines provide optimized operation for the Regex accelerator card. There are three String engines: String ICMP XL, String TCP XL, and String UDP XL.

Note The IPS 4345, IPS 4360, IPS 4510, IPS 4520, IPS 4520-XL, ASA 5525-X IPS SSP, ASA 5545-X IPS SSP, ASA 5555-X IPS SSP, and ASA 5585-X IPS SSP support the String XL engines and the Regex accelerator card.

Note The Regex accelerator card is used for both the standard String engines and the String XL engines. Most standard String engine signatures can be compiled and analyzed by the Regex accelerator card without modification. However, there are special circumstances in which the standard String engine signatures cannot be compiled for the Regex accelerator card. In these situations a new signature is written in a String XL engine using the specific parameters in the String XL engine that do compile on the Regex accelerator card. The new signature in the String XL engine obsoletes the original signature in the standard String engine.

- Sweep—Analyzes sweeps from a single host (ICMP and TCP), from destination ports (TCP and UDP), and multiple ports with RPC requests between two nodes. There are two Sweep engines: Sweep and Sweep Other TCP.

- Traffic Anomaly—Inspects TCP, UDP, and other traffic for worms.
- Traffic ICMP—Analyzes nonstandard protocols, such as TFN2K, LOKI, and DDOS. There are only two signatures with configurable parameters.
- Trojan—Analyzes traffic from nonstandard protocols, such as BO2K andTFN2K. There are three Trojan engines: Bo2k, Tfn2k, and UDP. There are no user-configurable parameters in these engines.

Master Engine

The Master engine provides structures and methods to the other engines and handles input from configuration and alert output. This section describes the Master engine, and contains the following topics:

- General Parameters, page B-4
- Alert Frequency, page B-6
- Event Actions, page B-7

General Parameters

The following parameters are part of the Master engine and apply to all signatures (if it makes sense for that signature engine). Table B-1 lists the general master engine parameters.

Table B-1 Master Engine Parameters

Parameter	Description	Value
Signature ID	Specifies the ID of this signature.	*number*
Sub Signature ID	Specifies the sub ID of this signature	*number*
Alert Severity	Specifies the severity of the alert: • Dangerous alert • Medium-level alert • Low-level alert • Informational alert	High Medium Low Informational (default)
Sig Fidelity Rating	Specifies the rating of the fidelity of this signature.	0 to 100 (default = 100)
Promiscuous Delta	Specifies the delta value used to determine the seriousness of the alert.	0 to 30 (default = 5)
Signature Name	Specifies the name of the signature.	*sig-name*
Alert Notes	Provides additional information about this signature that will be included in the alert message.	*alert-notes*
User Comments	Provides comments about this signature.	*comments*
Alert Traits	Specifies traits you want to document about this signature.	0 to 65335
Release	Provides the release in which the signature was most recently updated.	*release*

Table B-1 Master Engine Parameters (continued)

Parameter	Description	Value
Signature Creation Date	Specifies the date the signature was created.	—
Signature Type	Specifies the signature category.	Anomaly Component Exploit Other
Engine	Specifies the engine to which the signature belongs. **Note** The engine-specific parameters appear under the Engine category.	—
Event Count	Specifies the number of times an event must occur before an alert is generated.	1 to 65535 (default = 1)
Event Count Key	Specifies the storage type on which to count events for this signature: • Attacker address • Attacker and victim addresses • Attacker address and victim port • Victim address • Attacker and victim addresses and ports	Axxx AxBx Axxb xxBx AaBb
Specify Alert Interval {Yes | No}	Enables the alert interval: • Alert Interval—Specifies the time in seconds before the event count is reset.	2 to 1000
Status	Specifies whether the signature is enabled or disabled, active or retired.	Enabled | Retired {Yes | No}
Obsoletes	Indicates that a newer signature has disabled an older signature.	—
Vulnerable OS List	When combined with passive OS fingerprinting, it allows the IPS to determine if it is likely a given attack is relevant to the target system.	AIX BSD General OS HP-UX IOS IRIX Linux Mac OS Netware Other Solaris UNIX Windows Windows NT Windows NT/2K/XP
Mars Category {Yes | No}	Maps signatures to a MARS attack category.[1]	—

1. This is a static information category that you can set in the configuration and view in the alerts.Refer to the MARS documentation for more information.

Promiscuous Delta

The promiscuous delta lowers the risk rating of certain alerts in promiscuous mode. Because the sensor does not know the attributes of the target system and in promiscuous mode cannot deny packets, it is useful to lower the prioritization of promiscuous alerts (based on the lower risk rating) so the administrator can focus on investigating higher risk rating alerts. In inline mode, the sensor can deny the offending packets so that they never reach the target host, so it does not matter if the target was vulnerable. Because the attack was not allowed on the network, the IPS does not subtract from the risk rating value. Signatures that are not service, OS, or application-specific have 0 for the promiscuous delta. If the signature is specific to an OS, service, or application, it has a promiscuous delta of 5, 10, or 15 calculated from 5 points for each category.

Caution We recommend that you do NOT change the promiscuous delta setting for a signature.

Obsoletes

The Cisco signature team uses the obsoletes field to indicate obsoleted, older signatures that have been replaced by newer, better signatures, and to indicate disabled signatures in an engine when a better instance of that engine is available. For example, some String XL hardware-accelerated signatures now replace equivalent signatures that were defined in the String engine.

Vulnerable OS List

When you combine the vulnerable OS setting of a signature with passive OS fingerprinting, the IPS can determine if it is likely that a given attack is relevant to the target system. If the attack is found to be relevant, the risk rating value of the resulting alert receives a boost. If the relevancy is unknown, usually because there is no entry in the passive OS fingerprinting list, then no change is made to the risk rating. If there is a passive OS fingerprinting entry and it does not match the vulnerable OS setting of a signature, the risk rating value is decreased. The default value by which to increase or decrease the risk rating is +/- 10 points.

For More Information

- For more information about promiscuous mode, see Promiscuous Mode, page 5-10.
- For more information about passive OS fingerprinting, see Configuring OS Identifications, page 9-24.

Alert Frequency

The purpose of the alert frequency parameter is to reduce the volume of the alerts written to the Event Store to counter IDS DoS tools, such as stick. There are four modes: Fire All, Fire Once, Summarize, and Global Summarize. The summary mode is changed dynamically to adapt to the current alert volume. For example, you can configure the signature to Fire All, but after a certain threshold is reached, it starts summarizing.

Table B-2 lists the alert frequency parameters.

Table B-2 Master Engine Alert Frequency Parameters

Parameter	Description	Value
Summary Mode	Specifies the mode used for summarization: • Fire All—Fires an alert on all events. • Fire Once—Fires an alert only once. • Global Summarize—Summarizes an alert so that it only fires once regardless of how many attackers or victims. • Summarize—Summarizes alerts.	Fire All Fire Once Global Summarize Summarize
Specify Summary Threshold {Yes | No}	Enables summary threshold mode: • Summary Threshold—Specifies the threshold number of alerts to send a signature into summary mode. • Summary Interval—Specifies the time in seconds used in each summary alert.	0 to 65535 1 to 1000
Specify Global Summary Threshold {Yes | No}	Enables global summary threshold mode: • Global Summary Threshold—Specifies the threshold number of events to take alerts into global summary.	1 to 65535
Summary Key	Specifies the storage type on which to summarize this signature: • Attacker address • Attacker and victim addresses • Attacker address and victim port • Victim address • Attacker and victim addresses and ports	Axxx AxBx Axxb xxBx AaBb

Event Actions

Note Most of the following event actions belong to each signature engine unless they are not appropriate for that particular engine.

The Cisco IPS supports the following event actions. Most of the event actions belong to each signature engine unless they are not appropriate for that particular engine.

Alert and Log Actions

- Product Alert—Writes an alert to Event Store.
- Produce Verbose Alert—Includes an encoded dump (possibly truncated) of the offending packet in the alert.

- Log Attacker Packets—Starts IP logging of packets containing the attacker address and sends an alert.
- Log Victim Packets—Starts IP logging of packets containing the victim address and sends an alert.
- Log Attacker/Victim Pair Packets (inline mode only)—Starts IP logging of packets containing the attacker/victim address pair.
- Request SNMP Trap—Sends request to the NotificationApp to perform SNMP notification.

Deny Actions

- Deny Packet Inline (inline mode only)—Does not transmit this packet.

> **Note** You cannot delete the event action override for Deny Packet Inline because it is protected. If you do not want to use that override, disable it.

- Deny Connection Inline (inline mode only)—Does not transmit this packet and future packets on the TCP Flow.
- Deny Attacker Victim Pair Inline (inline mode only)—Does not transmit this packet and future packets on the attacker/victim address pair for a specified period of time.
- Deny Attacker Service Pair Inline (inline mode only)—Does not transmit this packet and future packets on the attacker address victim port pair for a specified period of time.
- Deny Attacker Inline (inline mode only)—Does not transmit this packet and future packets from the attacker address for a specified period of time.

> **Note** This is the most severe of the deny actions. It denies the current and future packets from a single attacker address. Each deny address times out for *X* seconds from the first event that caused the deny to start, where X is the amount of seconds that you configured. You can clear all denied attacker entries by choosing **Monitoring > Properties > Denied Attackers > Clear List**, which permits the addresses back on the network.

- Modify Packet Inline (inline mode only)—Modifies packet data to remove ambiguity about what the end point might do with the packet.

> **Note** The event action Modify Packet Inline is part of the Normalizer engine. It scrubs the packet and corrects irregular issues such as bad checksum, out of range values, and other RFC violations.

Other Actions

> **Note** IPv6 does not support the following event actions: Request Block Host, Request Block Connection, or Request Rate Limit.

- Request Block Connection—Requests the ARC to block this connection.
- Request Block Host—Requests the ARC to block this attacker host.
- Request Rate Limit—Requests the ARC to perform rate limiting.
- Reset TCP Connection—Sends TCP resets to hijack and terminate the TCP flow.

Regular Expression Syntax

Regular expressions (Regex) are a powerful and flexible notational language that allow you to describe text. In the context of pattern matching, regular expressions allow a succinct description of any arbitrary pattern.

Table B-3 lists the IPS signature Regex syntax.

Table B-3 Signature Regular Expression Syntax

Metacharacter	Name	Description
?	Question mark	Repeat 0 or 1 times.
*	Star, asterisk	Repeat 0 or more times.
+	Plus	Repeat 1 or more times.
{x}	Quantifier	Repeat exactly *X* times.
{x,}	Minimum quantifier	Repeat at least *X* times.
.	Dot	Any one character except new line (0x0A).
[abc]	Character class	Any character listed.
[^abc]	Negated character class	Any character not listed.
[a-z]	Character range class	Any character listed inclusively in the range.
( )	Parenthesis	Used to limit the scope of other metacharacters.
|	Alternation, or	Matches either expression it separates.
^	caret	The beginning of the line.
\char	Escaped character	When *char* is a metacharacter or not, matches the literal *char.*
char	Character	When char is not a metacharacter, matches the literal char.
\r	Carriage return	Matches the carriage return character (0x0D).
\n	New line	Matches the new line character (0x0A).
\t	Tab	Matches the tab character (0x09).
\f	Form feed	Matches the form feed character (0x0C).
\xNN	Escaped hexadecimal character	Matches character with the hexadecimal code 0xNN (0<=N<=F).
\NNN	Escaped octal character	Matches the character with the octal code NNN (0<=N<=8).

All repetition operators will match the shortest possible string as opposed to other operators that consume as much of the string as possible thus giving the longest string match.

Table B-4 lists examples of Regex patterns.

Table B-4 Regex Patterns

To Match	Regular Expression
Hacker	Hacker
Hacker or hacker	[Hh]acker
Variations of bananas, banananas, bananananas	ba(na)+s
foo and bar on the same line with anything except a new line between them	foo.*bar
Either foo or bar	foo|bar
Either moon or soon	(m|s)oon

AIC Engine

The Application Inspection and Control (AIC) engine inspects HTTP web traffic and enforces FTP commands. This section describes the AIC engine and its parameters, and contains the following topics:

- Understanding the AIC Engine, page B-10
- AIC Engine and Sensor Performance, page B-10
- AIC Engine Parameters, page B-11

Understanding the AIC Engine

AIC provides thorough analysis of web traffic. It provides granular control over HTTP sessions to prevent abuse of the HTTP protocol. It allows administrative control over applications, such as instant messaging and gotomypc, that try to tunnel over specified ports. Inspection and policy checks for P2P and instant messaging are possible if these applications are running over HTTP. AIC also provides a way to inspect FTP traffic and control the commands being issued. You can enable or disable the predefined signatures or you can create policies through custom signatures.

Note The AIC engines run when HTTP traffic is received on AIC web ports. If traffic is web traffic, but not received on the AIC web ports, the Service HTTP engine is executed. AIC inspection can be on any port if it is configured as an AIC web port and the traffic to be inspected is HTTP traffic.

AIC Engine and Sensor Performance

Application policy enforcement is a unique sensor feature. Rather than being based on traditional IPS technologies that inspect for exploits, vulnerabilities, and anomalies, AIC policy enforcement is designed to enforce HTTP and FTP service policies. The inspection work required for this policy enforcement is extreme compared with traditional IPS inspection work. A large performance penalty is associated with using this feature. When AIC is enabled, the overall bandwidth capacity of the sensor is reduced.

AIC policy enforcement is disabled in the IPS default configuration. If you want to activate AIC policy enforcement, we highly recommend that you carefully choose the exact policies of interest and disable those you do not need. Also, if your sensor is near its maximum inspection load capacity, we recommend that you not use this feature since it can oversubscribe the sensor. We recommend that you use the adaptive security appliance firewall to handle this type of policy enforcement.

AIC Engine Parameters

The AIC engines define signatures for deep inspection of web traffic. They also define signatures that authorize and enforce FTP commands. There are two AIC engines: AIC HTTP and AIC FTP. The AIC engines have the following features:

- Web traffic:
  - RFC compliance enforcement
  - HTTP request method authorization and enforcement
  - Response message validation
  - MIME type enforcement
  - Transfer encoding type validation
  - Content control based on message content and type of data being transferred
  - URI length enforcement
  - Message size enforcement according to policy configured and the header
  - Tunneling, P2P and instant messaging enforcement.

    This enforcement is done using regular expressions. There are predefined signature but you can expand the list.
- FTP traffic:
  - FTP command authorization and enforcement

Table B-5 lists the parameters that are specific to the AIC HTTP engine.

Table B-5 AIC HTTP Engine Parameters

Parameter	Description
Signature Type	Specifies the type of AIC signature.
Content Types	Specifies an AIC signature that deals with MIME types: • Define Content Type—Associates actions such as denying a specific MIME type (image/gif), defining a message-size violation, and determining that the MIME-type mentioned in the header and body do not match. • Define Recognized Content Types—Lists content types recognized by the sensor.
Define Web Traffic Policy	Specifies the action to take when noncompliant HTTP traffic is seen. Alarm on Non-HTTP Traffic {Yes | No} enables the signature. This signature is disabled by default.
Max Outstanding Requests Overrun	Specifies the maximum allowed HTTP requests per connection (1 to 16).

Table B-5 AIC HTTP Engine Parameters (continued)

Parameter	Description
Msg Body Pattern	Uses Regex to define signatures that look for specific patterns in the message body.
Request Methods	Specifies the AIC signature that allows actions to be associated with HTTP request methods: • Define Request Method—Specifies get, put, and so forth. • Recognized Request Methods—Lists methods recognized by the sensor.
Transfer Encoding	Specifies the AIC signature that deals with transfer encodings: • Define Transfer Encoding—Associates an action with each method, such as compress, chunked, and so forth. • Recognized Transfer Encodings—Lists methods recognized by the sensor. • Chunked Transfer Encoding—Specifies actions to be taken when a chunked encoding error is seen.

Table B-6 lists the parameters that are specific to the AIC FTP engine.

Table B-6 AIC FTP Engine Parameters

Parameter	Description
Signature Type	Specifies the type of AIC signature.
FTP Commands	Associates an action with an FTP command: • FTP Command—Lets you choose the FTP command you want to inspect.
Unrecognized FTP Command	Inspects unrecognized FTP commands.

For More Information

- For the procedures for configuring AIC engine signatures, see Configuring Application Policy, page 7-43.
- For an example of a custom AIC signature, see Tuning an AIC Signature, page 7-44.
- For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Atomic Engine

The Atomic engine contains signatures for simple, single packet conditions that cause alerts to be fired. This section describes the Atomic engine, and contains the following topics:

- Atomic ARP Engine, page B-13
- Atomic IP Advanced Engine, page B-14
- Atomic IP Engine, page B-25

- Atomic IPv6 Engine, page B-29

Atomic ARP Engine

The Atomic ARP engine defines basic Layer 2 ARP signatures and provides more advanced detection of the ARP spoof tools dsniff and ettercap.

Table B-7 lists the parameters that are specific to the Atomic ARP engine.

Table B-7 Atomic ARP Engine Parameters

Parameter	Description	Value
Specify ARP Operation {Yes | No}	(Optional) Enables ARP operation: • ARP Operation—Specifies the type of ARP operation to inspect.	0 to 65535
Specify Mac Flip Times {Yes | No}	(Optional) Enables MAC address flip times: • Mac Flip Times—Specifies how many times to flip the MAC address in the alert.	0 to 65535
Specify Request Inbalance {Yes | No}	(Optional) Enables request inbalance: • Request Inbalance—Specifies the value for firing an alert when there are this many more requests than replies on the IP address.	0 to 65535
Specify Type of ARP Sig {Yes | No}	(Optional) Enables the ARP signature type: • Type of ARP Sig—Specifies the type of ARP signatures you want to fire on: – Destination Broadcast—Fires an alert for this signature when it sees an ARP destination address of 255.255.255.255. – Same Source and Destination—Fires an alert for this signature when it sees an ARP destination address with the same source and destination MAC address – Source Broadcast (default)—Fires an alert for this signature when it sees an ARP source address of 255.255.255.255. – Source Multicast—Fires an alert for this signature when it sees an ARP source MAC address of 01:00:5e:(00-7f).	Dst Broadcast Same Src and Dst Src Broadcast Src Multicast
Storage Key	Specifies the type of address key used to store persistent data: • Attacker address • Attacker and victim addresses • Victim address • Global	Axxx AxBx xxBx xxxx

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Atomic IP Advanced Engine

The Atomic IP Advanced engine parses and interprets the IPv6 header and its extensions, the IPv4 header and its options, ICMP, ICMPv6, TCP, and UDP, and seeks out anomalies that indicate unusual activity.

Atomic IP Advanced engine signatures do the following:

- Inspect for anomalies in IP addresses, for example, spoofed addresses.
- Inspect for bad information in the length fields of the packet.
- Fire informational alerts about the packet.
- Fire higher severity alerts for the limited set of known vulnerabilities.
- Duplicate any IPv6-specific signatures in Engine Atomic IP that can also apply to IPv6.
- Provide default signatures for identifying tunneled traffic based on IP address, port, protocol, and limited information from the packet data.

Only the outermost IP tunnel is identified. When an IPv6 tunnel or IPv6 traffic inside of an IPv4 tunnel is detected, a signature fires an alert. All of the other IPv6 traffic in embedded tunnels is not inspected. The following tunneling methods are supported, but not individually detected. For example, ISATAP, 6to4, and manual IPv6 RFC 4213 tunnels all appear as IPv6 in IPv4, which is detected by signature 1007:

- ISATAP
- 6to4 (RFC 3056)
- Manually configured tunnels (RFC 4213)
- IPv6 over GRE
- Teredo (IPv6) inside UDP
- MPLS (unencrypted)
- IPv6 over IPv6

IPv6 supports the following:

- Denying by source IP address, destination IP address, or IP address pair
- Alerts
- Resetting the TCP connection
- Logging

Atomic IP Advanced Engine Restrictions

The Atomic IP Advanced engine contains the following restrictions:

- Cannot detect the Layer 4 field of the packets if the packets are fragmented so that the Layer 4 identifier does not appear in the first packet.
- Cannot detect Layer 4 attacks in flows with packets that are fragmented by IPv6 because there is no fragment reassembly.
- Cannot detect attacks with tunneled flows.
- Limited checks are provided for the fragmentation header.

- There is no support for IPv6 on the management (command and control) interface
- If there are illegal duplicate headers, a signature fires, but the individual headers cannot be separately inspected.
- Anomaly detection does not support IPv6 traffic; only IPv4 traffic is directed to the anomaly detection processor.
- Rate limiting and blocking are not supported for IPv6 traffic. If a signature is configured with a block or rate limit event action and is triggered by IPv6 traffic, an alert is generated but the action is not carried out.

Table B-8 on page B-15 lists the parameters that are specific to the Atomic IP Advanced engine.

Note The second number in the ranges must be greater than or equal to the first number.

Table B-8 Atomic IP Advanced Engine Parameters

Parameter	Description	Value
Global		
Fragment Status	Specifies whether or not fragments are wanted.	Any | No Fragments | Want Fragments
Encapsulation {Yes | No}	(Optional) Enables any encapsulation before the start of Layer 3 for the packet:[1] • Encapsulation—Specifies the type of encapsulation to inspect.	None | MPLS | GRE | Ipv4 in IPv6 | IP IP | Any
IP Version {Yes | No}	(Optional) Enables the IP protocol version: • IP Version—Specifies IPv4 or IPv6.	IPv4 | IPv6
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)
Regex		
Specify Regex Inspection	(Optional) Enables Regex inspection.	Yes | No

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Regex Search Scope	Specifies the start and end points for the regular expression search.	ipv6-doh-only ipv6-doh-plus ipv6-hoh-only ipv6-hoh-plus ipv6-rh-only ipv6-rh-plus layer3-only layer3-plus layer4
Regex String	Specifies the regular expression to search for in a single TCP packet.	*string*
Specify Exact Match Offset {Yes | No}	Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the Regex String must match.	0 to 65535
Specify Min Match Offset {Yes | No}	Enables minimum match offset: • Min Match Offset—Specifies the minimum stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Max Match Offset {Yes | No}	Enables maximum match offset: • Max Match Offset—Specifies the maximum stream offset the Regex String must report for a match to be valid.	0 to 65535
IPv6		

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Authentication Header {Yes | No}	(Optional) Enables inspection of the authentication header: • AH Present—Inspects the authentication header: – AH Length—Specifies the length of the authentication header to inspect. – AH Next Header—Specifies the value of the authentication header to inspect.	Have AH | No AH • 0 to 1028 • 0 to 255
Destination Options Header {Yes | No}	(Optional) Enables inspection of the destination options header: • DOH Present —Inspects the destination options header: – DOH Count —Specifies the number of destination options headers to inspect. – DOH Length—Specifies the length of destination options headers to inspect. – DOH Next Header—Specifies the number of next destination options headers to inspect. – DOH Option Type—Specifies the type of destination options headers to inspect. – DOH Option Length —Specifies the length of destination options headers to inspect.	Have DOH | No DO • 0 to 2 • 8 to 2048 • 0 to 255 • 0 to 255 • 0 to 255
Specify ESP Header {Yes | No}	(Optional) Enables inspection of the ESP header: • ESP Present—Inspects the ESP header.	Have ESP | No ESP
Specify First Next Header {Yes | No}	(Optional) Enables inspection of the first next header: • First Next Header—Specifies the value of the first next header to inspect.	0 to 255

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify Flow Label {Yes | No}	(Optional) Enables inspection of the flow label: • Flow Label—Specifies the value of the flow label to inspect.	0 to 1048575
Specify Headers Out of Order {Yes | No}	(Optional) Enables inspection of out-of-order headers: • Headers Out of Order—Inspects headers that are out of order.	Yes | No
Specify Headers Repeated {Yes | No}	(Optional) Enables inspection of repeated headers: • Headers Repeated—Inspects repeated headers.	Yes | No
Specify Hop Limit {Yes | No}	(Optional) Enables hop limit: • Hop Limit—Specifies the value of the hop limit to inspect.	0 to 255
Specify Hop Options Header {Yes | No}	(Optional) Enables inspection of the hop-by-hop options header: • HOH Present—Inspects the hop-by-hop options header.	Have HOH | No HOH

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify IPv6 Address Options {Yes | No}	(Optional) Enables the IPv6 address options: • IPv6 Address Options—Specifies the IPv6 address options: – Address With Localhost—IP address with ::1. – Documentation Address—IP address with 2001:db8::/32 prefix. – IPv6 Address—IP address. – Link Local Address—Inspects for an IPv6 link local address. – Multicast Destination—Inspects for a destination multicast address. – Multicast Source—Inspects for a source multicast address. – Not Link Local Address—Inspects for an address that is not link-local. – Not Valid Address—Inspects for an address that is not reserved for link-local, global, or multicast. – Source IP Equals Destination IP—Source and destination addresses are the same.	Yes | No
Specify IPv6 Data Length {Yes | No}	(Optional) Enables inspection of IPv6 data length: • IPv6 Data Length—Specifies the IPv6 data length to inspect.	0 to 65535
Specify IPv6 Header Length {Yes | No}	(Optional) Enables inspection of IPv6 header length: • IPv6 Header Length—Specifies the length of the IPv6 header to inspect.	0 to 65535
Specify IPv6 Total Length {Yes | No}	(Optional) Enables inspection of IPv6 total length: • IPv6 Total Length—Specifies the IPv6 total length to inspect.	0 to 65535

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify IPv6 Payload Length {Yes | No}	(Optional) Enables inspection of IPv6 payload length: • IPv6 Payload Length—Specifies the IPv6 payload length to inspect.	0 to 65535
Specify Routing Header {Yes | No}	(Optional) Enables inspection of the routing header: • RH Present—Inspects the routing header.	Have RH | No RH
Specify Traffic Class {Yes | No}	(Optional) Enables inspection of the traffic class: • Traffic Class—Specifies the value of the traffic class to inspect.	0 to 255
IPV4		
Specify IP Addr Options {Yes | No}	(Optional) Enables IP address options: • IP Addr Options—Specifies the IP address options.	Address With Localhost P Address[2] RFC 1918 Address Src IP Eq Dst IP
Specify IP Header Length {Yes | No}	(Optional) Enables inspection of the IP header length: • IP Header Length—Specifies the length of the IP header to inspect.	0 to 16
Specify IP Identifier {Yes | No}	(Optional) Enables inspection of the IP identifier: • IP Identifier—Specifies the IP ID to inspect.	0 to 255
Specify IP Option Inspection {Yes | No}	(Optional) Enables inspection of the IP options: • IP Option Inspection—Specifies the value of the IP option: – IP Option—IP OPTION code to match. – IP Option Abnormal Options—The list of options is malformed.	0 to 65535

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify IP Payload Length {Yes | No}	(Optional) Enables inspection of the IP payload length: • IP Payload Length—Specifies the length of the IP payload to inspect.	0 to 65535
Specify IP Type of Service {Yes | No}	(Optional) Enables inspection of the IP type of service: • IP Type of Service—Specifies the IP type of service to inspect.	0 to 255
Specify IP Total Length {Yes | No}	(Optional) Enables inspection of the IP total length: • IP Total Length—Specifies the total length of the IP packet to inspect.	0 to 65535
Specify IP Time-to-Live {Yes | No}	(Optional) Enables inspection of the IP time-to-live: • IP Time-to-Live—Specifies the value of the IP TTL to inspect.	0 to 255
Specify IP Version {Yes | No}	(Optional) Enables inspection of the IP version: • IP Version—Specifies which IP version to inspect.	0 to 16
L4 Protocol		
Specify L4 Protocol {Yes | No}	(Optional) Enables inspection of Layer 4 protocol: • L4 Protocol—Specifies which Layer 4 protocol to inspect.	ICMP Protocol ICMPv6 Protocol TCP Protocol UDP Protocol Other IP Protocols
L4 Protocol Other		
Specify Other IP Protocol ID	(Optional) Enables inspection of other Layer 4 protocols: • Other IP Protocol ID—Specifies which single IP protocol ID or single range of IP protocol IDs for which to send alerts.	0 to 255
L4 Protocol ICMP		

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify ICMP Code {Yes | No}	(Optional) Enables inspection of Layer 4 ICMP code: • ICMP Code—Specifies the value of the ICMP header CODE.	0 to 65535
Specify ICMP ID {Yes | No}	(Optional) Enables inspection of Layer 4 ICMP ID: • ICMP ID—Specifies the value of the ICMP header IDENTIFIER.	0 to 65535
Specify ICMP Sequence {Yes | No}	(Optional) Enables inspection of Layer 4 ICMP sequence: • ICMP Sequence—Specifies the ICMP sequence for which to look.	0 to 65535
Specify ICMP Type {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMP header type: • ICMP Type—Specifies the value of the ICMP header TYPE.	0 to 65535
L4 Protocol ICMPv6		
Specify ICMPv6 Code	(Optional) Enables inspection of the Layer 4 ICMPv6 code: • ICMPv6 Code—Specifies the value of the ICMPv6 header CODE.	0 to 255
Specify ICMPv6 ID {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 identifier: • ICMPv6 ID—Specifies the value of the ICMPv6 header IDENTIFIER.	0 to 65535
Specify ICMPv6 Length {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 length: • ICMPv6 Length—Specifies the value of the ICMPv6 header LENGTH.	0 to 65535
Specify ICMPv6 MTU Field {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 MTU field: • ICMPv6 MTU Field—Specifies the value of the ICMPv6 header MTU field.	4,294,967,295
Specify ICMPv6 Option Type {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 type: • ICMPv6 Option Type—Specifies the ICMPv6 option type to inspect.	0 to 255

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify ICMPv6 Option Length {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 option length: • ICMPv6 Option Length—Specifies the ICMPv6 option length to inspect.	0 to 255
Specify ICMPv6 Sequence {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 sequence: • ICMPv6 Sequence—Specifies the value of the ICMPv6 header SEQUENCE.	0 to 65535
Specify ICMPv6 Type {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMPv6 type: • ICMPv6 Type—Specifies the value of the ICMPv6 header TYPE.	0 to 255
L4 Protocol TCP and UDP		
Specify Destination Port {Yes | No}	(Optional) Enables the destination port for use: • Destination Port—Specifies the destination port of interest for this signature.	0 to 65535
Specify Source Port {Yes | No}	(Optional) Enables source port for use: • Source Port—Specifies the source port of interest for this signature.	0 to 65535
Specify TCP Mask {Yes | No}	(Optional) Enables the TCP mask for use: • TCP Mask—Specifies the mask used in TCP flags comparison: – URG bit – ACK bit – PSH bit – RST bit – SYN bit – FIN bit	URG ACK PSH RST SYN FIN

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify TCP Flags {Yes | No}	(Optional) Enables TCP flags for use: • TCP Flags—Specifies the TCP flags to match when masked by mask: – URG bit – ACK bit – PSH bit – RST bit – SYN bit – FIN bit	URG ACK PSH RST SYN FIN
Specify TCP Reserved {Yes | No}	(Optional) Enables TCP reserved for use: • TCP Reserved—Specifies the value of TCP reserved.	0 to 63
Specify TCP Header Length {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP header length: • TCP Header Length—Specifies the length of the TCP header used in inspection.	0 to 60
Specify TCP Payload Length {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP payload length: • TCP Payload Length—Specifies the length of the TCP payload.	0 to 65535
Specify TCP URG Pointer {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP URG pointer: • TCP URG Pointer—Specifies the value of the TCP URG flag inspection.	0 to 65535
Specify TCP Window Size {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP window size: • TCP Window Size—Specifies the window size of the TCP packet.	0 to 65535

Table B-8 Atomic IP Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify UDP Valid Length {Yes | No}	(Optional) Enables inspection of the Layer 4 UDP valid length: • UDP Valid Length—Specifies the UDP packet lengths that are considered valid and should not be inspected.	0 to 65535
Specify UDP Length Mismatch {Yes | No}	(Optional) Enables inspection of the Layer 4 UDP length mismatch: • UDP Length Mismatch—Fires an alert when IP Data length is less than the UDP Header length.	0 to 65535

1. When a packet is GRE, IPIP, IPv4inIPv6, or MPL the sensor skips the Layer 3 encapsulation header and the encapsulation header, and all inspection is done starting from the second Layer 3. The encapsulation enumerator allows the engine to look backward to see if there is an encapsulation header before the Layer 3 in question.
2. Use the following syntax: x.x.x.x-z.z.z.z, for example, 10.10.10.1-10.10.10.254.

For More Information

- For an example custom Atomic IP Advanced Engine signature, see Example Atomic IP Advanced Engine Signature, page 7-25.
- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Atomic IP Engine

The Atomic IP engine defines signatures that inspect IP protocol headers and associated Layer 4 transport protocols (TCP, UDP, and ICMP) and payloads. The Atomic engines do not store persistent data across packets. Instead they can fire an alert from the analysis of a single packet.

Table B-9 lists the parameters that are specific to the Atomic IP engine.

Table B-9 Atomic IP Engine Parameters

Parameter	Description	Value
Specify IP Addr Options {Yes | No}	(Optional) Enables IP address options: • IP Addr Options—Specifies the IP address options.	Address With Localhost IP Address[1] RFC 1918 Address Src IP Eq Dst IP
Specify IP Header Length {Yes | No}	(Optional) Enables inspection of the IP header length: • IP Header Length—Specifies the length of the IP header to inspect.	0 to 16

Table B-9 Atomic IP Engine Parameters (continued)

Parameter	Description	Value
Specify IP Identifier {Yes | No}	(Optional) Enables inspection of the IP identifier: • IP Identifier—Specifies the IP ID to inspect.	0 to 255
Specify IP Option Inspection {Yes | No}	(Optional) Enables inspection of the IP options: • IP Option Inspection—Specifies the value of the IP option: – IP Option—Specifies the IP OPTION code to match. – IP Option Abnormal Options—Specifies the list of options is malformed.	0 to 65535
Specify IP Payload Length {Yes | No}	(Optional) Enables inspection of the IP payload length: • IP Payload Length—Specifies the length of IP payload to inspect.	0 to 65535
Specify IP Type of Service {Yes | No}	(Optional) Specifies the IP type of service: • IP Type of Service—Specifies the IP type of service to inspect.	0 to 6 255
Specify IP Total Length {Yes | No}	(Optional) Enables inspection of the IP total length: • IP Total Length—Specifies the total length of IP packet to inspect.	0 to 65535
Specify IP Time-to-Live {Yes | No}	(Optional) Enables inspection of IP time-to-live: • IP Time-to-Live—Specifies the value of the IP TTL to inspect.	0 to 255
Specify IP Version {Yes | No}	(Optional) Enables inspection of the IP version: • IP Version—Specifies which IP version to inspect.	0 to 16
Specify L4 Protocol {Yes | No}	(Optional) Enables inspection of the Layer 4 protocol: • L4 Protocol—Specifies which Layer 4 protocol to inspect.	ICMP Protocol TCP Protocol UDP Protocol Other IP Protocols
Specify ICMP Code {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMP code: • ICMP Code—Specifies the value of the ICMP header CODE.	0 to 65535

Table B-9 Atomic IP Engine Parameters (continued)

Parameter	Description	Value
Specify ICMP ID {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMP ID: • ICMP ID—Specifies the value of the ICMP header IDENTIFIER.	0 to 65535
Specify ICMP Sequence {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMP sequence: • ICMP Sequence—Specifies the ICMP sequence to inspect.	0 to 65535
Specify ICMP Type {Yes | No}	(Optional) Enables inspection of the ICMP header type: • ICMP Type—Specifies the value of the ICMP header TYPE.	0 to 65535
Specify ICMP Total Length {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMP total header length: • ICMP Total Length—Specifies the value of the ICMP total length to inspect.	0 to 65535
Specify Other IP Protocol ID {Yes | No}	(Optional) Enables inspection of the other Layer 4 protocols: • Other IP Protocol ID—Specifies which single IP protocol ID or single range of IP protocol IDs for which to send alerts.	0 to 255
Specify Destination Port {Yes | No}	(Optional) Enables the destination port for use: • Destination Port—Specifies the destination port of interest for this signature.	0 to 65535
Specify Source Port {Yes | No}	(Optional) Enables source port for use: • Source Port—Specifies the source port of interest for this signature.	0 to 65535
Specify TCP Mask {Yes | No}	(Optional) Enables the TCP mask for use: • TCP Mask—Specifies the mask used in TCP flags comparison: – URG bit – ACK bit – PSH bit – RST bit – SYN bit – FIN bit	URG ACK PSH RST SYN FIN

Table B-9 Atomic IP Engine Parameters (continued)

Parameter	Description	Value
Specify TCP Flags {Yes | No}	(Optional) Enables TCP flags for use: • TCP Flags—Specifies the TCP flags to match when masked by mask: – URG bit – ACK bit – PSH bit – RST bit – SYN bit – FIN bit	URG ACK PSH RST SYN FIN
Specify TCP Reserved {Yes | No}	(Optional) Enables TCP reserved for use: • TCP Reserved—Specifies the value of TCP reserved.	0 to 63
Specify TCP Header Length {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP header length: • TCP Header Length—Specifies the length of the TCP header used in inspection.	0 to 60
Specify TCP Payload Length {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP payload length: • TCP Payload Length—Specifies the length of the TCP payload.	0 to 65535
Specify TCP URG Pointer {Yes | No}	(Optional) Enables inspection of the L4 TCP URG pointer: • TCP URG Pointer—Specifies the value of the TCP URG flag to inspect.	0 to 65535
Specify TCP Window Size {Yes | No}	(Optional) Enables inspection of the Layer 4 TCP window size: • TCP Window Size—Specifies the window size of the TCP packet.	0 to 65535
Specify UDP Length {Yes | No}	(Optional) Enables inspection of the Layer 4 UDP length: • UDP Length—Fires an alert when the IP Data length is less than the UDP Header length.	0 to 65535

Table B-9 Atomic IP Engine Parameters (continued)

Parameter	Description	Value
Specify UDP Valid Length {Yes | No}	(Optional) Enables inspection of the Layer 4 UDP valid length: • UDP Valid Length—Specifies UDP packet lengths that are considered valid and should not be inspected.	0 to 65535
Specify UDP Length Mismatch {Yes | No}	(Optional) Enables inspection of the Layer 4 UDP length mismatch: • UDP Length Mismatch—Fires an alert when the IP Data length is less than the UDP Header length.	0 to 65535

1. Use the following syntax: x.x.x.x-z.z.z.z, for example, 10.10.10.1-10.10.10.254.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Atomic IPv6 Engine

The Atomic IPv6 engine detects two IOS vulnerabilities that are stimulated by malformed IPv6 traffic. These vulnerabilities can lead to router crashes and other security issues. One IOS vulnerability deals with multiple first fragments, which cause a buffer overflow. The other one deals with malformed ICMPv6 Neighborhood Discovery options, which also cause a buffer overflow.

Note IPv6 increases the IP address size from 32 bits to 128 bits, which supports more levels of addressing hierarchy, a much greater number of addressable nodes, and autoconfiguration of addresses.

Atomic IPv6 Signatures

There are eight Atomic IPv6 signatures. The Atomic IPv6 inspects Neighborhood Discovery protocol of the following types:

- Type 133—Router Solicitation
- Type 134—Router Advertisement
- Type 135—Neighbor Solicitation
- Type 136—Neighbor Advertisement
- Type 137—Redirect

Note Hosts and routers use Neighborhood Discovery to determine the link-layer addresses for neighbors known to reside on attached links and to quickly purge cached values that become invalid. Hosts also use Neighborhood Discovery to find neighboring routers that will forward packets on their behalf.

Each Neighborhood Discovery type can have one or more Neighborhood Discovery options. The Atomic IPv6 engine inspects the length of each option for compliance with the legal values stated in RFC 2461. Violations of the length of an option results in an alert corresponding to the option type where the malformed length was encountered (signatures 1601 to 1605).

Note The Atomic IPv6 signatures do not have any specific parameters to configure.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Fixed Engine

The Fixed engines combine multiple regular expression patterns in to a single pattern matching table that allows a single search through the data. It supports ICMP, TCP, and UDP protocols. After a minimum inspection depth is reached (1 to 100 bytes), inspection stops. There are three Fixed engines: Fixed ICMP, Fixed TCP, and Fixed UDP.

Note The Fixed TCP and Fixed UDP engines use the Service Ports parameter as exclusion ports. The Fixed ICMP engine uses the Service Ports parameter as excluded ICMP types.

Table B-10 lists the parameters specific to the Fixed ICMP engine.

Table B-10 Fixed ICMP Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
Max Payload Inspect Length	Specifies the maximum inspection depth for the signature.	1 to 250
Regex String	Specifies the regular expression to search for in a single packet.	*string*
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the Regex String must report for a match to be valid.	0 to 65535

Table B-10 Fixed ICMP Engine Parameters (continued)

Parameter	Description	Value
Specify Minimum Match Length {Yes | No}	(Optional) Enables minimum match length: - Minimum Match Length—Specifies the minimum number of bytes the Regex String must match.	0 to 65535
Specify ICMP Type {Yes | No}	(Optional) Enables inspection of the Layer 4 ICMP header type: - ICMP Type—Specifies the value of the ICMP header TYPE.	0 to 65535
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

Table B-11 lists the parameters specific to the Fixed TCP engine.

Table B-11 Fixed TCP Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of traffic: - Traffic from service port destined to client port. - Traffic from client port destined to service port.	From Service To Service
Max Payload Inspect Length	Specifies the maximum inspection depth for the signature.	1 to 250
Regex String	Specifies the regular expression to search for in a single packet.	*string*
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: - Exact Match Offset—Specifies the exact stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: - Min Match Length—Specifies the minimum number of bytes the Regex String must match.	0 to 65535

Table B-11 Fixed TCP Engine Parameters (continued)

Parameter	Description	Value
Exclude Service Ports {Yes | No}	Enables service ports for use: - Excluded Service Ports—Specifies a comma-separated list of ports or port ranges to exclude.	0 to 65535[1] a-b[,c-d]
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

Table B-12 lists the parameters specific to the Fixed UDP engine.

Table B-12 Fixed UDP Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of traffic: - Traffic from service port destined to client port. - Traffic from client port destined to service port	From Service To Service
Max Payload Inspect Length	Specifies the maximum inspection depth for the signature.	1 to 250
Regex String	Specifies the regular expression to search for in a single packet.	*string*
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: - Exact Match Offset—Specifies the exact stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: - Min Match Length—Specifies the minimum number of bytes the Regex String must match.	0 to 65535

Table B-12 Fixed UDP Engine Parameters (continued)

Parameter	Description	Value
Exclude Service Ports {Yes | No}	Enables service ports for use: • Excluded Service Ports—Specifies a comma-separated list of ports or port ranges to exclude.	0 to 65535[1] a-b[,c-d]
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Flood Engine

The Flood engines define signatures that watch for any host or network sending multiple packets to a single host or network. For example, you can create a signature that fires when 150 or more packets per second (of the specific type) are found going to the victim host. There are two Flood engines: Flood Host and Flood Net.

Table B-13 lists the parameters specific to the Flood Host engine.

Table B-13 Flood Host Engine Parameters

Parameter	Description	Value
Protocol	Specifies which kind of traffic to inspect.	ICMP UDP
Rate	Specifies the threshold number of packets per second.	0 to 65535[1]
ICMP Type	Specifies the value for the ICMP header type.	0 to 65535
Dst Ports	Specifies the destination ports when you choose UDP protocol.	0 to 65535[2] a-b[,c-d]
Src Ports	Specifies the source ports when you choose UDP protocol.	0 to 65535[2] a-b[,c-d]

[1] An alert fires when the rate is greater than the packets per second.

[2] The second number in the range must be greater than or equal to the first number.

Table B-14lists the parameters specific to the Flood Net engine.

Table B-14 Flood Net Engine Parameters

Parameter	Description	Value
Gap	Specifies the gap of time allowed (in seconds) for a flood signature.	0 to 65535
Peaks	Specifies the number of allowed peaks of flood traffic.	0 to 65535
Protocol	Specifies which kind of traffic to inspect.	ICMP TCP UDP
Rate	Specifies the threshold number of packets per second.	0 to 65535[1]
Sampling Interval	Specifies the interval used for sampling traffic.	1 to 3600
ICMP Type	Specifies the value for the ICMP header type.	0 to 65535

[1] An alert fires when the rate is greater than the packets per second.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Meta Engine

Caution A large number of Meta engine signatures could adversely affect overall sensor performance.

The Meta engine defines events that occur in a related manner within a sliding time interval. This engine processes events rather than packets. As signature events are generated, the Meta engine inspects them to determine if they match any or several Meta definitions. The Meta engine generates a signature event after all requirements for the event are met.

All signature events are handed off to the Meta engine by the Signature Event Action Processor. The Signature Event Action Processor hands off the event after processing the minimum hits option. Summarization and event action are processed after the Meta engine has processed the component events.

Table B-15 lists the parameters specific to the Meta engine.

Table B-15 Meta Engine Parameters

Parameter	Description	Value
Component List	Specifies the Meta engine component: • edit—Edits an existing entry. • insert—Inserts a new entry into the list: – begin—Places the entry at the beginning of the active list. – end—Places the entry at the end of the active list. – inactive—Places the entry into the inactive list. – before—Places the entry before the specified entry. – after—Places the entry after the specified entry. • move—Moves an entry in the list. • Component Count—Specifies the number of times component must fire before this component is satisfied • Component Sig ID—Specifies the signature ID of the signature to match this component on. • Component SubSig ID—Specifies the subsignature ID of the signature to match this component on. • Is Not Component—Specifies that the component is a NOT component.	*name1* Yes | No
Component List In Order	Specifies to fire the component list in order.	Yes | No
All Components Required	Specifies to use all components.	Yes | No
All NOT Components Required	Specifies to use all of the NOT components.	Yes | No
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)
Meta Reset Interval	Specifies the time in seconds to reset the Meta signature.	0 to 3600

Table B-15 Meta Engine Parameters (continued)

Parameter	Description	Value
Meta Key	Specifies the storage type for the Meta signature: • Attacker address • Attacker and victim addresses • Attacker and victim addresses and ports • Victim address	• Axxx • AxBx • AaBb • xxBx
Unique Victims	Specifies the number of unique victims ports required per Meta signature.	1 to 256

For More Information

- For an example of a custom Meta engine signature, see Example Meta Engine Signature, page 7-21.
- For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Multi String Engine

Caution The Multi String engine can have a significant impact on memory usage.

The Multi String engine lets you define signatures that inspect Layer 4 transport protocol (ICMP, TCP, and UDP) payloads using multiple string matches for one signature. You can specify a series of regular expression patterns that must be matched to fire the signature. For example, you can define a signature that looks for regex 1 followed by regex 2 on a UDP service. For UDP and TCP you can specify port numbers and direction. You can specify a single source port, a single destination port, or both ports. The string matching takes place in both directions.

Use the Multi String engine when you need to specify more than one Regex pattern. Otherwise, you can use the String ICMP, String TCP, or String UDP engine to specify a single Regex pattern for one of those protocols.

Table B-16lists the parameters specific to the Multi String Engine.

Table B-16 Multi String Engine Parameters

Parameter	Description	Value
Inspect Length	Specifies the length of the stream or packet that must contain all offending strings for the signature to fire.	0 to 4294967295
Protocol	Specifies the Layer 4 protocol selection.	ICMP TCP UDP

Table B-16 Multi String Engine Parameters (continued)

Parameter	Description	Value
Regex Component	Specifies the list of Regex components: • Regex String—Specifies the string to search for. • Spacing Type—Specifies the type of spacing required from the match before or from the beginning of the stream/packet if it is the first entry in the list.	list (1 to 16 items) exact minimum
Port Selection	Specifies the type of TCP or UDP port to inspect: • Both Ports—Specifies both source and destination port. • Destination—Specifies a range of destination ports. • Source—Specifies a range of source ports.[1]	0 to 65535 [2]
Extra Spacing	Specifies the exact number of bytes that must be between this Regex string and the one before, or from the beginning of the stream/packet if it is the first entry in the list.	0 to 4294967296
Minimum Spacing	Specifies the minimum number of bytes that must be between this Regex string and the one before, or from the beginning of the stream/packet if it is the first entry in the list.	0 to 4294967296
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] Port matching is performed bidirectionally for both the client-to-server and server-to-client traffic flow directions. For example, if the source-ports value is 80, in a client-to-server traffic flow direction, inspection occurs if the client port is 80. In a server-to-client traffic flow direction, inspection occurs if the server port is port 80.

[2] A valid value is a comma- separated list of integer ranges a-b[,c-d] within 0 to 65535. The second number in the range must be greater than or equal to the first number.

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Normalizer Engine

Note You cannot add custom signatures to the Normalizer engine. You can tune the existing ones.

The Normalizer engine deals with IP fragment reassembly and TCP stream reassembly. With the Normalizer engine you can set limits on system resource usage, for example, the maximum number of fragments the sensor tries to track at the same time. Sensors in promiscuous mode report alerts on violations. Sensors in inline mode perform the action specified in the event action parameter, such as Produce Alert, Deny Packet Inline, and Modify Packet Inline.

Caution For signature 3050 Half Open SYN Attack, if you choose Modify Packet Inline as the action, you can see as much as 20 to 30% performance degradation while the protection is active. The protection is only active during an actual SYN flood.

IP Fragmentation Normalization

Intentional or unintentional fragmentation of IP datagrams can hide exploits making them difficult or impossible to detect. Fragmentation can also be used to circumvent access control policies like those found on firewalls and routers. And different operating systems use different methods to queue and dispatch fragmented datagrams. If the sensor has to check for all possible ways that the end host can reassemble the datagrams, the sensor becomes vulnerable to DoS attacks. Reassembling all fragmented datagrams inline and only forwarding completed datagrams and refragmenting the datagram if necessary, prevents this. The IP Fragmentation Normalization unit performs this function.

TCP Normalization

Through intentional or natural TCP session segmentation, some classes of attacks can be hidden. To make sure policy enforcement can occur with no false positives and false negatives, the state of the two TCP endpoints must be tracked and only the data that is actually processed by the real host endpoints should be passed on. Overlaps in a TCP stream can occur, but are extremely rare except for TCP segment retransmits. Overwrites in the TCP session should not occur. If overwrites do occur, someone is intentionally trying to elude the security policy or the TCP stack implementation is broken. Maintaining full information about the state of both endpoints is not possible unless the sensor acts as a TCP proxy. Instead of the sensor acting as a TCP proxy, the segments are ordered properly and the Normalizer engine looks for any abnormal packets associated with evasion and attacks.

IPv6 Fragments

The Normalizer engine can reassemble IPv6 fragments and forward the reassembled buffer for inspection and actions by other engines and processors. The following differences exist between IPv4 and IPv6:

- Modify Packet Inline for Normalizer engine signatures has no effect on IPv6 datagrams.
- Signature 1206 (IP Fragment Too Small) does not fire for IPv6 datagrams. Signature 1741 in the Atomic IP Advanced engine fires for IPv6 fragments that are too small.
- Signature 1202 allows 48 additional bytes beyond the Maximum Datagram Size for IPv6 because of the longer IPv6 header fields.

TCP Normalizer Signature Warning

You receive the following warning if you disable a default-enabled TCP Normalizer signature or remove a default-enabled modify packet inline, deny packet inline, or deny connection inline action:

```
Use caution when disabling, retiring, or changing the event action settings of a <Sig ID>
TCP Normalizer signature for a sensor operating in IPS mode. The TCP Normalizer signature
default values are essential for proper operation of the sensor.
If the sensor is seeing duplicate packets, consider assigning the traffic to multiple
virtual sensors. If you are having problems with asymmetric or out-of-order TCP packets,
consider changing the normalizer mode from strict evasion protection to asymmetric mode
protection. Contact Cisco TAC if you require further assistance.
```

ASA IPS Modules and the Normalizer Engine

The majority of the features in the Normalizer engine are not used on the ASA 5500-X IPS SSP or ASA 5585-X IPS SSP, because the ASA itself handles the normalization. Packets on the ASA IPS modules go through a special path in the Normalizer that only reassembles fragments and puts packets in the right order for the TCP stream. The Normalizer does not do any of the normalization that is done on an inline IPS appliance, because that causes problems in the way the ASA handles the packets.

The following Normalizer engine signatures are not supported:

- 1300.0
- 1304.0
- 1305.0
- 1307.0
- 1308.0
- 1309.0
- 1311.0
- 1315.0
- 1316.0
- 1317.0
- 1330.0
- 1330.1
- 1330.2
- 1330.9
- 1330.10
- 1330.12
- 1330.14
- 1330.15
- 1330.16
- 1330.17
- 1330.18

Table B-17 lists the parameters that are specific to the Normalizer engine.

Table B-17 Normalizer Engine Parameters

Parameter	Description
Edit Defaults	Editable signatures.
Specify Fragment Reassembly Timeout	(Optional) Enables fragment reassembly timeout.
Specify Hijack Max Old Ack	(Optional) Enables hijack-max-old-ack.
Specify Max Datagram Size	(Optional) Enables maximum datagram size.
Specify Max Fragments	(Optional) Enables maximum fragments: • Max Fragments—Lets you specify the number of maximum fragments.

Table B-17 Normalizer Engine Parameters (continued)

Parameter	Description
Specify Max Fragments per Datagram	(Optional) Enables maximum fragments per datagram.
Specify Max Last Fragments	(Optional) Enables maximum last fragments.
Specify Max Partial Datagrams	(Optional) Enables maximum partial datagrams.
Specify Max Small Frags	(Optional) Enables maximum small fragments.
Specify Min Fragment Size	(Optional) Enables minimum fragment size.
Specify Service Ports	(Optional) Enables service ports.
Specify SYN Flood Max Embryonic	(Optional) Enables SYN flood maximum embryonic.
Specify TCP Closed Timeout	(Optional) Enables TCP closed timeout.
Specify TCP Embryonic Timeout	(Optional) Enables TCP embryonic timeout.
Specify TCP Idle Timeout	(Optional) Enables TCP idle timeout: • TCP Idle Timeout—Lets you specify the TCP idle timeout time.
Specify TCP Max MSS	(Optional) Enables TCP maximum mss.
Specify TCP Max Queue	(Optional) Enables TCP maximum queue.
Specify TCP Min MSS	(Optional) Enables TCP minimum mss.
Specify TCP Option Number	(Optional) Enables TCP option number.

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For the procedures for configuring signatures in the Normalizer engine, see Configuring IP Fragment Reassembly Signatures, page 7-45, and Configuring TCP Stream Reassembly Signatures, page 7-48.

Service Engines

This section describes the Service engines, and contains the following topics:

- Service P2P, page B-54
- Service RPC Engine, page B-54
- Service SMB Advanced Engine, page B-56
- Service SNMP Engine, page B-58
- Service SSH Engine, page B-59
- Service TNS Engine, page B-59

Understanding the Service Engines

The Service engines analyze Layer 5+ traffic between two hosts. These are one-to-one signatures that track persistent data. The engines analyze the Layer 5+ payload in a manner similar to the live service.

The Service engines have common characteristics but each engine has specific knowledge of the service that it is inspecting. The Service engines supplement the capabilities of the generic string engine specializing in algorithms where using the string engine is inadequate or undesirable.

Service DNS Engine

The Service DNS engine specializes in advanced DNS decode, which includes anti-evasive techniques, such as following multiple jumps. It has many parameters, such as lengths, opcodes, strings, and so forth. The Service DNS engine is a biprotocol inspector operating on both TCP and UDP port 53. It uses the stream for TCP and the quad for UDP.

Table B-18 lists the parameters specific to the Service DNS engine.

Table B-18 Service DNS Engine Parameters

Parameter	Description	Value
Protocol	Specifies the protocol of interest for this inspector.	TCP UDP
Specify Query Chaos String {Yes | No}	(Optional) Enables the DNS Query Class Chaos String: • Query Chaos String—Specifies the query chaos string to search on.	*query-chaos-string*
Specify Query Class {Yes | No}	(Optional) Enables the query class: • Query Class—Specifies the DNS Query Class 2 Byte Value.	0 to 65535
Specify Query Invalid Domain Name {Yes | No}	(Optional) Enables the query invalid domain name: • Query Invalid Domain Name—Specifies the DNS Query Length greater than 255.	Yes | No
Specify Query Jump Count Exceeded {Yes | No}	(Optional) Enables query jump count exceeded: • Query Jump Count Exceeded—DNS compression counter.	Yes | No

Table B-18 Service DNS Engine Parameters (continued)

Parameter	Description	Value
Specify Query Opcode {Yes | No}	(Optional) Enables query opcode: • Query Opcode—Specifies the DNS Query Opcode 1 byte Value.	0 to 65535
Specify Query Record Data Invalid {Yes | No}	(Optional) Enables query record data invalid: • Query Record Data Invalid—Specifies the DNS Record Data incomplete.	Yes | No
Specify Query Record Data Length {Yes | No}	(Optional) Enables the query record data length: • Query Record Data Length—Specifies the DNS Response Record Data Length.	0 to 65535
Specify Query Src Port 53 {Yes | No}	(Optional) Enables the query source port 53: • Query Src Port 53—Specifies the DNS packet source port 53.	Yes | No
Specify Query Stream Length {Yes | No}	(Optional) Enables the query stream length: • Query Record Data Length—Specifies the DNS Packet Length.	0 to 65535
Specify Query Type {Yes | No}	(Optional) Enables the query type: • Query Type—Specifies the DNS Query Type 2 Byte Value.	0 to 65535
Specify Query Value {Yes | No}	(Optional) Enables the query value: • Query Value—Specifies the Query 0 Response 1.	Yes | No

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service FTP Engine

The Service FTP engine specializes in FTP port command decode, trapping invalid **port** commands and the PASV port spoof. It fills in the gaps when the String engine is not appropriate for detection. The parameters are Boolean and map to the various error trap conditions in the **port** command decode. The Service FTP engine runs on TCP ports 20 and 21. Port 20 is for data and the Service FTP engine does not do any inspection on this. It inspects the control transactions on port 21.

Table B-19 lists the parameters that are specific to the Service FTP engine.

Table B-19 Service FTP Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
FTP Inspection Type	Specifies the type of inspection to perform: • Looks for an invalid address in the FTP port command. • Looks for an invalid port in the FTP port command. • Looks for the PASV port spoof.	Invalid Address in PORT Command Invalid Port in PORT Command PASV Port Spoof
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service Generic Engine

The Service Generic engine allows programmatic signatures to be issued in a config-file-only signature update. It has a simple machine and assembly language that is defined in the configuration file. It runs the machine code (distilled from the assembly language) through its virtual machine, which processes the instructions and pulls the important pieces of information out of the packet and runs them through the comparisons and operations specified in the machine code. It is intended as a rapid signature response engine to supplement the String and State engines.

New functionality adds the Regex parameter to the Service Generic engine and enhanced instructions. The Service Generic engine can analyze traffic based on the mini-programs that are written to parse the packets. These mini-programs are composed of commands, which dissect the packet and look for certain conditions.

Note You cannot use the Service Generic engine to create custom signatures.

Caution Due to the proprietary nature of this complex language, we do not recommend that you edit the Service Generic engine signature parameters other than severity and event action.

Table B-20 lists the parameters specific to the Service Generic engine.

Table B-20 Service Generic Engine Parameters

Parameter	Description	Value
Specify Dst Port {Yes | No}	(Optional) Enables the destination port: • Dst Port—Specifies the destination port of interest for this signature.	0 to 65535
Specify IP Protocol {Yes | No}	(Optional) Enables IP protocol: • IP Protocol—Specifies the IP protocol this inspector should examine.	0 to 255
Specify Payload Source {Yes | No}	(Optional) Enables payload source inspection: • Payload Source—Specifies the payload source inspection for the following types: – Inspects ICMP data – Inspects Layer 2 headers – Inspects Layer 3 headers – Inspects Layer 4 headers – Inspects TCP data – Inspects UDP data	ICMP Data 12 Header 13 Header 14 Header TCP Data UDP Data
Specify Src Port {Yes | No}	(Optional) Enables the source port: • Src Port—Specifies the source port of interest for this signature.	0 to 65535
Specify Regex String {Yes | No}	Specifies the regular expression to look for when the policy type is Regex: • Regex String—Specifies a regular expression to search for in a single TCP packet. • (Optional) Specify Min Match Length—Enables minimum match length for use: – Min Match Length—Specifies the minimum length of the Regex match required to constitute a match.	*string* 0 to 65535
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Service H225 Engine

The Service H225 engine analyzes H225.0 protocol, which consists of many subprotocols and is part of the H.323 suite. H.323 is a collection of protocols and other standards that together enable conferencing over packet-based networks.

H.225.0 call signaling and status messages are part of the H.323 call setup. Various H.323 entities in a network, such as the gatekeeper and endpoint terminals, run implementations of the H.225.0 protocol stack. The Service H225 engine analyzes H225.0 protocol for attacks on multiple H.323 gatekeepers, VoIP gateways, and endpoint terminals. It provides deep packet inspection for call signaling messages that are exchanged over TCP PDUs. The Service H225 engine analyzes the H.225.0 protocol for invalid H.255.0 messages, and misuse and overflow attacks on various protocol fields in these messages.

H.225.0 call signaling messages are based on Q.931 protocol. The calling endpoint sends a Q.931 setup message to the endpoint that it wants to call, the address of which it procures from the admissions procedure or some lookup means. The called endpoint either accepts the connection by transmitting a Q.931 connect message or rejects the connection. When the H.225.0 connection is established, either the caller or the called endpoint provides an H.245 address, which is used to establish the control protocol (H.245) channel.

Especially important is the SETUP call signaling message because this is the first message exchanged between H.323 entities as part of the call setup. The SETUP message uses many of the commonly found fields in the call signaling messages, and implementations that are exposed to probable attacks will mostly also fail the security checks for the SETUP messages. Therefore, it is highly important to check the H.225.0 SETUP message for validity and enforce checks on the perimeter of the network.

The Service H225 engine has built-in signatures for TPKT validation, Q.931 protocol validation, and ASN.1PER validations for the H225 SETUP message. ASN.1 is a notation for describing data structures. PER uses a different style of encoding. It specializes the encoding based on the data type to generate much more compact representations.

You can tune the Q.931 and TPKT length signatures and you can add and apply granular signatures on specific H.225 protocol fields and apply multiple pattern search signatures of a single field in Q.931 or H.225 protocol.

The Service H225 engine supports the following features:

- TPKT validation and length check
- Q.931 information element validation
- Regular expression signatures on text fields in Q.931 information elements
- Length checking on Q.931 information elements
- SETUP message validation
- ASN.1 PER encode error checks
- Configuration signatures for fields like ULR-ID, E-mail-ID, h323-id, and so forth for both regular expression and length.

There is a fixed number of TPKT and ASN.1 signatures. You cannot create custom signatures for these types. For TPKT signatures, you should only change the value-range for length signatures. You should not change any parameters for ASN.1. For Q.931 signatures, you can add new regular expression signatures for text fields. For SETUP signatures, you can add signatures for length and regular expression checks on various SETUP message fields.

Table B-21 lists parameters specific to the Service H225 engine.

Table B-21 Service H.225 Engine Parameters

Parameter	Description	Value
Message Type	Specifies the type of H225 message to which the signature applies: • SETUP • ASN.1-PER • Q.931 • TPKT	ASN.1-PER Q.931 SETUP TPKT
Policy Type	Specifies the type of H225 policy to which the signature applies: • Inspects field length. • Inspects presence. If certain fields are present in the message, an alert is sent. • Inspects regular expressions. • Inspects field validations. • Inspects values. **Note** Regex and presence are not valid for TPKT signatures.	Field Validation Length Check Presence Regex Value
Specify Field Name {Yes | No}	(Optional) Enables field name for use. Gives a dotted representation of the field name to which this signature applies. • Field Name—Specifies the field name to inspect. **Note** Only valid for SETUP and Q.931 message types.	1 to 512
Specify Invalid Packet Index {Yes | No}	(Optional) Enables invalid packet index for use for specific errors in ASN, TPKT, and other errors that have fixed mapping. • Invalid Packet Index—Specifies the inspection for invalid packet index.	0 to 255

Table B-21 Service H.225 Engine Parameters (continued)

Parameter	Description	Value
Value Range Regex String {Yes | No}	Specifies the regular expression to look for when the policy type is Regex: • Regex String—Specifies a regular expression to search for in a single TCP packet. • (Optional) Specify Min Match Length—Enables minimum match length for use: – Min Match Length—Specifies the minimum length of the Regex match required to constitute a match. **Note** This is never set for TPKT signatures.	*string* 0 to 65535
Specify Value Range {Yes | No}	Enables value range for use: • Value Range—Specifies the range of values. **Note** Valid for the length or value policy types (0x00 to 6535). Not valid for other policy types.	0 to 65535[1] a-b
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Service HTTP Engine

The Service HTTP engine is a service-specific string-based pattern-matching inspection engine. The HTTP protocol is one of the most commonly used in networks of today. In addition, it requires the most amount of preprocessing time and has the most number of signatures requiring inspection making it critical to the overall performance of the system.

The Service HTTP engine uses a Regex library that can combine multiple patterns into a single pattern-matching table allowing a single search through the data. This engine searches traffic directed only to web services, or HTTP requests. You cannot inspect return traffic with this engine. You can specify separate web ports of interest in each signature in this engine.

HTTP deobfuscation is the process of decoding an HTTP message by normalizing encoded characters to ASCII equivalent characters. It is also known as ASCII normalization.

Before an HTTP packet can be inspected, the data must be deobfuscated or normalized to the same representation that the target system sees when it processes the data. It is ideal to have a customized decoding technique for each host target type, which involves knowing what operating system and web server version is running on the target. The Service HTTP engine has default deobfuscation behavior for the Microsoft IIS web server.

Table B-22 lists the parameters specific the Service HTTP engine.

Table B-22 Service HTTP Engine Parameters

Parameter	Description	Value
De Obfuscate	Applies anti-evasive deobfuscation before searching.	Yes | No
Max Field Sizes	Enables maximum field sizes grouping.	—
Specify Max Arg Field Length {Yes | No}	(Optional) Enables maximum argument field length: • Max Arg Field Length—Specifies the maximum length of the arguments field.	0 to 65535
Specify Max Header Field Length {Yes | No}	(Optional) Enables maximum header field length: • Max Header Field Length—Specifies the maximum length of the header field.	0 to 65535
Specify Max Request Field Length {Yes | No}	(Optional) Enables maximum request field length: • Max Request Field Length—Specifies the maximum length of the request field.	0 to 65535
Specify Max URI Field Length {Yes | No}	(Optional) Enables the maximum URI field length: • Max URI Field Length—Specifies the maximum length of the URI field.	0 to 65535
Regex	Enables regular expression grouping.	—
Specify Arg Name Regex {Yes | No}	(Optional) Enables searching the Arguments field for a specific regular expression: • Arg Name Regex—Specifies the regular expression to search for in the HTTP Arguments field (after the ? and in the Entity body as defined by Content-Length).	—
Specify Header Regex {Yes | No}	(Optional) Enables searching the Header field for a specific regular expression: • Header Regex—Specifies the regular expression to search in the HTTP Header field. **Note** The Header is defined after the first CRLF and continues until CRLFCRLF.	—

Table B-22 Service HTTP Engine Parameters (continued)

Parameter	Description	Value
Specify Request Regex {Yes | No}	(Optional) Enables searching the Request field for a specific regular expression: • Request Regex—Specifies the regular expression to search in both HTTP URI and HTTP Argument fields. • Specify Min Request Match Length—Enables setting a minimum request match length: – Min Request Match Length—Specifies the minimum request match length.	0 to 65535
Specify URI Regex {Yes | No}	(Optional) Specifies the regular expression to search in HTTP URI field. **Note** The URI field is defined to be after the HTTP method (GET, for example) and before the first CRLF. **Note** The regular expression is protected, which means you cannot change the value.	[/\\][a-zA-Z][a-zA-Z][a-zA-Z][a-zA-Z][a-zA-Z][a-zA-Z][a-zA-Z][.]jpeg
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No default)

[1] The second number in the range must be greater than or equal to the first number.

For More Information

- For an example Service HTTP custom signature, see Example Service HTTP Engine Signature, page 8-17.
- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Service IDENT Engine

The Service IDENT engine inspects TCP port 113 traffic. It has basic decode and provides parameters to specify length overflows. For example, when a user or program at computer A makes an IDENT request of computer B, it may only ask for the identity of users of connections between A and B. The IDENT server on B listens for connections on TCP port 113. The client at A establishes a connection, then specifies which connection it wants identification for by sending the numbers of the ports on A and B that the connection is using. The server at B determines what user is using that connection, and replies to A with a string that names that user. The Service IDENT engine inspects the TCP port 113 for IDENT abuse.

Table B-23 lists the parameters specific to the Service IDENT engine.

Table B-23 Service IDENT Engine Parameters

Parameter	Description	Value
Inspection Type	Specifies the type of inspection to perform.	Has Newline Has Bad Port Payload Size
Has Newline	Inspects payload for a nonterminating new line character.	—
Has Bad Port	Inspects payload for a bad port.	—
Payload Size	Inspects for payload length longer than this: • Max Bytes—Specifies the maximum bytes for the payload length.	0 to 65535
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]
Direction	Specifies the direction of the traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service

[1] The second number in the range must be greater than or equal to the first number.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4

Service MSRPC Engine

The Service MSRPC engine processes MSRPC packets. MSRPC allows for cooperative processing between multiple computers and their application software in a networked environment. It is a transaction-based protocol, implying that there is a sequence of communications that establishes the channel and passes processing requests and replies.

MSRPC is an ISO Layer 5-6 protocol and is layered on top of other transport protocols such as UDP, TCP, and SMB. The MSRPC engine contains facilities to allow for fragmentation and reassembly of the MSRPC PDUs.

This communication channel is the source of recent Windows NT, Windows 2000, and Window XP security vulnerabilities. The Service MSRPC engine only decodes the DCE and RPC protocol for the most common transaction types.

Table B-24 lists the parameters specific to the Service MSRPC engine.

Table B-24 Service MSRPC Engine Parameters

Parameter	Description	Value
Protocol	Enables the protocol of interest for this inspector: • Type—Specifies UDP or TCP.	TCP UDP
Specify Flags {Yes | No}	Enables the flags to set: • MSRPC TCP Flags—Specifies MSRPC TCP flags. • MSRPC TCP Flags Mask—Specifies the MSRPC TCP flags mask.	Concurrent Execution Did Not Execute First Fragment Last Fragment Maybe Semantics Object UUID Pending Cancel Reserved
Specify Operation {Yes | No}	(Optional) Enables using MSRPC operation: • Operation—Specifies the MSRPC operation requested. **Note** Required for SMB_COM_TRANSACTION commands. Exact match.	0 to 65535
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

Table B-24 Service MSRPC Engine Parameters (continued)

Parameter	Description	Value
Specify Regex String {Yes | No}	(Optional) Enables using a regular expression string: • Specify Exact Match Offset—Enables the exact match offset: – Exact Match Offset—Specifies the exact stream offset the regular expression string must report for a match to be valid. • Specify Min Match Length—Enables the minimum match length: – Min Match Length—Specifies the minimum number of bytes the regular expression string must match. • Specify Min Match Offset—Enables the minimum match length: – Min Match Offset—Specifies the minimum stream offset the regular expression string must report for a match to be valid. • Specify Max Match Offset—Enables the maximum match offset: – Max Match Offset—Specifies the maximum stream offset the regular expression string must report for a match to be valid.	0 to 65535
Specify UUID {Yes | No}	(Optional) Enables UUID: • UUID—Specifies the MSRPC UUID field.	000001a000000000c000000000000046

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Service MSSQL Engine

The Service MSSQL engine inspects the protocol used by the Microsoft SQL server. There is one MSSQL signature. It fires an alert when it detects an attempt to log in to an MSSQL server with the default sa account. You can add custom signatures based on MSSQL protocol values, such as login username and whether a password was used.

Table B-25 lists the parameters specific to the Service MSSQL engine.

Table B-25 Service MSSQL Engine Parameters

Parameter	Description	Value
Password Present	Specifies whether or not a password was used in an MS SQL login.	Yes | No
Specify SQL Username	(Optional) Enables using an SQL username: • SQL Username—Specifies the username (exact match) of user logging in to MS SQL service.	sa

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service NTP Engine

The Service NTP engine inspects NTP protocol. There is one NTP signature, the NTP readvar overflow signature, which fires an alert if a readvar command is seen with NTP data that is too large for the NTP service to capture. You can tune this signature and create custom signatures based on NTP protocol values, such as mode and size of control packets.

Table B-26 lists the parameters specific to the Service NTP engine.

Table B-26 Service NTP Engine Parameters

Parameter	Description	Value
Inspection Type	Specifies the type of inspection to perform.	Inspect NTP Packets Is Invalid Data Packet Is Non NTP Traffic
Inspect NTP Packets	Enables inspection of NTP packets: • Control Opcode—Specifies the opcode number of an NTP control packet according to RFC1305, Appendix B. • Max Control Data Size—Specifies the maximum allowed amount of data sent in a control packet. • Operation Mode—Specifies the mode of operation of the NTP packet per RFC 1305.	0 to 65535
Is Invalid Data Packet	Enables inspection of invalid NTP data packets and checks the structure of the NTP data packet to make sure it is the correct size.	—
Is Non NTP Traffic	Enables the inspection of nonNTP packets on an NTP port.	—

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service P2P

P2P networks use nodes that can simultaneously function as both client and server for the purpose of file sharing. P2P networks often contain copyrighted material and their use on a corporate network can violate company policy. The Service P2P engine monitors such networks and provides optimized TCP and UDP P2P protocol identification. The Service P2P engine has the following characteristics:

- Listens on all TCP and UDP ports.
- Increased performance through the use of hard-coded signatures rather than regular expressions.
- Ignores traffic once P2P protocol is identified or after seeing 10 packets without a P2P protocol being identified.

Note Because the P2P signatures are hard coded, the only parameters that you can edit are the Master engine parameters.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service RPC Engine

The Service RPC engine specializes in RPC protocol and has full decode as an anti-evasive strategy. It can handle fragmented messages (one message in several packets) and batch messages (several messages in a single packet).

The RPC portmapper operates on port 111. Regular RPC messages can be on any port greater than 550. RPC sweeps are like TCP port sweeps, except that they only count unique ports when a valid RPC message is sent. RPC also runs on UDP.

Table B-27 lists the parameters specific to the Service RPC engine.

Table B-27 Service RPC Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
Protocol	Specifies the protocol of interest.	TC UDP
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]

Table B-27 Service RPC Engine Parameters (continued)

Parameter	Description	Value
Specify Regex String {Yes | No}	(Optional) Enables using a regular expression string: • Specify Exact Match Offset—Enables the exact match offset: – Exact Match Offset—Specifies the exact stream offset the regular expression string must report for a match to be valid. • Specify Min Match Length—Enables the minimum match length: – Min Match Length—Specifies the minimum number of bytes the regular expression string must match.	0 to 65535
Specify Spoof Src {Yes | No}	(Optional) Enables the spoof source address: • Is Spoof Src—Fires an alert when the source address is 127.0.0.1.	Yes | No
Specify Port Map Program {Yes | No}	(Optional) Enables the portmapper program: • Port Map Program—Specifies the program number sent to the portmapper for this signature.	0 to 9999999999
Specify RPC Max Length {Yes | No}	(Optional) Enables RPC maximum length: • RPC Max Length—Specifies the maximum allowed length of the entire RPC message. **Note** Lengths longer than what you specify fire an alert.	0 to 65535
Specify RPC Procedure {Yes | No}	(Optional) Enables RPC procedure: • RPC Procedure—Specifies the RPC procedure number for this signature.	0 to 1000000
Specify RPC Program {Yes | No}	(Optional) Enables RPC program: • RPC Program—Specifies the RPC program number for this signature.	0 to 1000000
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Service SMB Advanced Engine

Note The SMB engine has been replaced by the SMB Advanced engine. Even though the SMB engine is still visible in IDM, IME, and the CLI, its signatures have been obsoleted; that is, the new signatures have the obsoletes parameter set with the IDs of their corresponding old signatures. Use the new SMB Advanced engine to rewrite any custom signature that were in the SMB engine.

The Service SMB Advanced engine processes Microsoft SMB and Microsoft RPC over SMB packets. The Service SMB Advanced engine uses the same decoding method for connection-oriented MSRPC as the MSRPC engine with the requirement that the MSRPC packet must be over the SMB protocol. The Service SMB Advanced engine supports MSRPC over SMB on TCP ports 139 and 445. It uses a copy of the connection-oriented DCS/RPC code from the MSRPC engine.

Table B-28 lists the parameters specific to the Service SMB Advanced engine.

Table B-28 Service SMB Advanced Engine Parameters

Parameter	Description	Value
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535 a-b[,c-d] [1]
Specify SMB Command {Yes | No}	(Optional) Enables SMB commands: • SMB Command—Specifies the SMB command value. **Note** Exact match required; defines the SMB packet type.[2]	0 to 255
Specify Direction {Yes | No}	(Optional) Enables traffic direction: • Direction—Specifies the direction of traffic: – From Service—Traffic from service port destined to client port. – To Service—Traffic from client port destined to service port.	From Service To Service
Specify MSRPC over SMB Operation {Yes | No}	(Optional) Enables MSRPC over SMB: • MSRPC over SMB Operation—Specifies MSRPC over SMB. **Note** Required for SMB_COM_TRANSACTION commands, exact match required.	0 to 65535
Specify Regex String {Yes | No}	(Optional) Enables searching for Regex strings: • Regex String—Specifies a regular expression to search for in a single TCP packet.	*string*

Table B-28 Service SMB Advanced Engine Parameters (continued)

Parameter	Description	Value
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the Regex string must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the Regex string must match.	0 to 65535
Specify Regex Payload Source {Yes | No}	(Optional) Enables payload source inspection: • Payload Source—Specifies the kind of payload source inspection.[3]	Resource SMB Data TCP Data
Specify Scan Interval {Yes | No}	(Optional) Enables scan interval: • Scan Interval—Specifies the interval in seconds used to calculate alert rates.	1 to 131071
Specify TCP Flags {Yes | No}	(Optional) Enables TCP flags: • MSRPC TCP Flags—Specifies the MSRPC TCP flags. • MSRPC TCP Flags Mask—Specifies the MSRPC flags mask.	Concurrent Execution Did Not Execute First Fragment Last Fragment Maybe Semantics Object UUID Pending Cancel Reserved
Specify MSRPC over SMB PDU Type {Yes | No}	(Optional) Enables MSRPC PDU type over the SMB packet: • MSRPC over SMB PDU Type—Specifies the PDU type of MSRPC over the SMB packet.	0 = Request 2 = Response 11 = Bind 12 = Bind Ack
Specify MSRPC over SMB UUID {Yes | No}	(Optional) Enables MSRPC over UUID: • MSRPC over SMB UUID—Specifies the MSRPC UUID.	32-character string composed of hexadecimal characters 0-9, a-f, A-F.
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	True | False (default)

[1] The second number in the range must be greater than or equal to the first number.

[2] Currently supporting 37 (0x25) SMB_COM_TRANSACTION command \x26amp; 162 (0xA2) SMB_COM_NT_CREATE_ANDX command.

[3] TCP_Data performs Regex over entire packet, SMB_Data performs Regex on SMB payload only, Resource_DATA performs Regex on SMB_Resource.

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

Service SNMP Engine

The Service SNMP engine inspects all SNMP packets destined for port 161. You can tune SNMP signatures and create custom SNMP signatures based on specific community names and object identifiers.

Instead of using string comparison or regular expression operations to match the community name and object identifier, all comparisons are made using the integers to speed up the protocol decode and reduce storage requirements.

Table B-29 lists the parameters specific to the Service SNMP engine.

Table B-29 Service SNMP Engine Parameters

Parameter	Description	Value
Inspection Type	Enables the SNMP inspection type.	Brute Force Inspection (default) Invalid Packet Inspection Non-SNMP Traffic Inspection SNMP Inspection
Brute Force Inspection	Enables brute force inspection: • Bruce Force Count—Specifies the number of unique SNMP community names that constitute a brute force attempt.	0 to 65535
Invalid Packet Inspection	Inspects for SNMP protocol violations.	—
Non-SNMP Traffic Inspection	Inspects for non-SNMP traffic destined for UDP port 161.	—
SNMP Inspection {Yes | No}	Enables inspection of SNMP traffic: • Specify Object ID—Enables inspection of the SNMP Object identifier: – Object ID—Specifies to search for the SNMP object identifier. • Specify Community Name—Enables inspection of the SNMP community name: – Community Name—Specifies to search for the SNMP community name (SNMP password).	*object-id* *community-name*

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service SSH Engine

The Service SSH engine specializes in port 22 SSH traffic. Because all but the setup of an SSH session is encrypted, the Service SSH engine only looks at the fields in the setup. There are two default signatures for SSH. You can tune these signatures, but you cannot create custom signatures.

Table B-30 lists the parameters specific to the Service SSH engine.

Table B-30 Service SSH Engine Parameters

Parameter	Description	Value
Length Type	Inspects for one of the following SSH length types: • Key Length—Enables inspection of the length of the SSH key: – Length—Specifies that keys larger than this fire the RSAREF overflow. • User Length—Enables user length SSH inspection: – Length—Specifies that keys larger than this fire the RSAREF overflow.	0 to 65535
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]
Specify Packet Depth {Yes | No}	(Optional) Enables packet depth: • Packet Depth—Specifies the number of packets to watch before determining the session key was missed.	0 to 65535

[1] The second number in the range must be greater than or equal to the first number.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Service TNS Engine

The Service TNS engine inspects TNS protocol. TNS provides database applications with a single common interface to all industry-standard network protocols. With TNS, applications can connect to other database applications across networks with different protocols. The default TNS listener port is TCP 1521. TNS also supports REDIRECT frames that redirect the client to another host and/or another TCP port. To support REDIRECT packets, the TNS engine listens on all TCP ports and has a quick TNS frame header validation routine to ignore non-TNS streams.

Table B-31 lists the parameters specific to the Service TNS engine

.

Table B-31 Service TNS Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
Specify Regex String {Yes | No}	(Optional) Enables using a regular expression string: • Regex String—Specifies the regular expression to search for.	*string*
Specify Exact Match Offset {Yes | No}	Enables the exact match offset: • Exact Match Offset—Specifies the exact stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Max Match Offset {Yes | No}	Enables maximum match offset: • Max Match Offset—Specifies the maximum stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Min Match Offset {Yes | No}	Enables minimum match offset: • Min Match Offset—Specifies the minimum stream offset the Regex String must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	Enables the minimum match length: – Min Match Length—Specifies the minimum number of bytes the Regex String must match.	0 to 65535
Specify Regex Payload Source {Yes | No}	Enables the inspection of TCP or TNS protocol: • Payload Source—Specifies which protocol to inspect: – TCP Data—Performs Regex over the data portion of the TCP packet. – TNS Data—Performs Regex only over the TNS data (with all white space removed).	TCP Data TNS Data
Type Frame Type	Specifies the TNS frame value type: • 1—Connect • 2—Accept • 4—Refuse • 5—Redirect • 6—Data • 11—Resend • 12—Marker	1 2 4 5 6 11 12

For More Information

- For more information on the parameters common to all signature engines, see Master Engine, page B-4.
- For a list of the signature regular expression syntax, see Regular Expression Syntax, page B-9.

State Engine

The State engine provides state-based regular expression-based pattern inspection of TCP streams. A state engine is a device that stores the state of an event and at a given time can operate on input to transition from one state to another and/or cause an action or output to take place. State machines are used to describe a specific event that causes an output or alarm. There are three state machines in the State engine: SMTP, Cisco Login, and LPR Format String.

Table B-32 lists the parameters specific to the State engine.

Table B-32 State Engine Parameters

Parameter	Description	Value
State Machine	Specifies the state machine grouping.	Cisco Login LPR Format String SMPT
Cisco Login	Specifies the state machine for Cisco login: • State Name—Name of the state required before the signature fires an alert: – Cisco device state – Control-C state – Password prompt state – Start state	Cisco Device Control C Pass Prompt Start
LPR Format String	Specifies the state machine to inspect for the LPR format string vulnerability: • State Name—Name of the state required before the signature fires an alert: – Abort state to end LPR Format String inspection – Format character state – State state	Abort Format Char Start

Table B-32 State Engine Parameters (continued)

Parameter	Description	Value
SMPT	Specifies the state machine for the SMTP protocol: • State Name—Name of the state required before the signature fires an alert: – Abort state to end LPR Format String inspection – Mail body state – Mail header state – SMTP commands state – Start state	Abort Mail Body Mail Header SMPT Commands Start
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the regular expression string must match.	0 to 65535
Regex String	Specifies the regular expression to search for. **Note** This parameter is protected; you cannot edit it.	*string*
Direction	Specifies the direction of the traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the regular expression string must report for a match to be valid.	0 to 65535
Specify Max Match Offset {Yes | No}	(Optional) Enables maximum match offset: • Max Match Offset—Specifies the maximum stream offset the regular expression string must report for a match to be valid.	0 to 65535
Specify Min Match Offset {Yes | No}	(Optional) Enables minimum match offset: • Min Match Offset—Specifies the minimum stream offset the regular expression string must report for a match to be valid.	0 to 65535

[1] The second number in the range must be greater than or equal to the first number.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

String Engines

The String engine is a generic-based pattern-matching inspection engine for ICMP, TCP, and UDP protocols. The String engine uses a regular expression engine that can combine multiple patterns into a single pattern-matching table allowing for a single search through the data. There are three String engines: String ICMP, String TCP, and String UDP.

Table B-33 lists the parameters specific to the String ICMP engine.

Table B-33 String ICMP Engine Parameters

Parameter	Description	Value
Direction	Specifies the direction of the traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
ICMP Type	Specifies the value of the ICMP header TYPE.	0 to 18[1] a-b[,c-d]
Regex String	The Regex pattern to use in the search.	string
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the regular expression string must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the regular expression string must match.	0 to 65535
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

Table B-34 lists the parameters specific to the String TCP engine.

Table B-34 String TCP Engine

Parameter	Description	Value
Direction	Specifies the direction of the traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
Regex String	The Regex pattern to use in the search.	*string*
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]

Table B-34 String TCP Engine (continued)

Parameter	Description	Value
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the regular expression string must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the regular expression string must match.	0 to 65535
Strip Telnet Options	Strips the Telnet option characters from the data before the pattern is searched.[2]	Yes | No
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)

[1] The second number in the range must be greater than or equal to the first number.

[2] This parameter is primarily used as an IPS anti-evasion tool.

Table B-35 lists the parameters specific to the String UDP engine.

Table B-35 String UDP Engine

Parameter	Description	Value
Direction	Specifies the direction of the traffic: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	• From Service • To Service
Regex String	The Regex pattern to use in the search.	*string*
Service Ports	Specifies a comma-separated list of ports or port ranges where the target service resides.	0 to 65535[1] a-b[,c-d]
Specify Exact Match Offset {Yes | No}	(Optional) Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset the regular expression string must report for a match to be valid.	0 to 65535
Specify Min Match Length {Yes | No}	(Optional) Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the regular expression string must match.	0 to 65535
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No

[1] The second number in the range must be greater than or equal to the first number.

For More Information

- For an example custom String engine signature, see Example String TCP Engine Signature, page 8-22.
- For more information on the parameters common to all signature engines, see Master Engine, page B-4.

String XL Engines

Note The IPS 4345, IPS 4360, IPS 4510, IPS 4520, ASA 5525-X IPS SSP, ASA 5545-X IPS SSP, ASA 5555-X IPS SSP, and ASA 5585-X IPS SSP support the String XL engines and the Regex accelerator card.

The String XL engines do the same thing as the other String engines—provide a matching capability of one string per signature—but they use a different Regex syntax.The String TCP XL engine is stream-based and uses cross-packet inspection (XPI). The packets must be in order. UDP and ICMP are both stateless, thus the String UDP XL and String ICMP XL signature engines require no session state to be allocated and so each packet is a separate search.

The Regex accelerator card is used for both the standard String engines and the String XL engines. Most standard String engine signatures can be compiled and analyzed by the Regex accelerator card without modification. However, there are special circumstances in which the standard String engine signatures cannot be compiled for the Regex accelerator card. In these situations a new signature is written in a String XL engine using the specific parameters in the String XL engine that do compile on the Regex accelerator card. The new signature in the String XL engine obsoletes the original signature in the standard String engine.

Although you can use regular expression syntax or raw expression syntax, raw expression syntax is for expert users only. When configuring String XL signatures, the Regex String parameter is required unless you are using raw expression syntax.

Note Raw Regex is regular expression syntax used for raw mode processing. It is expert mode only and targeted for use by the Cisco IPS signature development team or only those who are under supervision by the Cisco IPS signature development team. You can configure a String XL signature in either regular Regex or raw Regex.

Table B-36 lists the parameters specific to the String XL engines (TCP, ICMP, and UDP).

Table B-36 String XL Engine Parameters

Parameter	Description	Value
Direction	(Required) Direction of the traffic to inspect: • Traffic from service port destined to client port. • Traffic from client port destined to service port.	From Service To Service
Dot All	If set to Yes, matches [\x00-\xFF] including \n; if set to No, matches anything in the range [\x00-\xFF] except \n.	Yes | No (default)
End Optional	Specifies that at the end of a packet, if all other conditions are satisfied but the end is not seen, a match is reported if the minimum is exceeded.	Yes | No (default)
ICMP Type	Specifies the ICMP message type. Required if the signature engine is String ICMP.	0 to 18[1] a-b[,c-d]
No Case	Specifies to treat all alphabetic characters in the expression as case insensitive.	Yes | No (default)
Raw Regex	If set to Yes, Min Match Length, Max Match Length, Min Whole Length, Max Whole Length, Dot All, UTF8, No Case, Stingy, and End Optional are not used to reformat the regular expression string. **Note** Raw Regex lets you enter a regular expression string in Raw syntax without being translated.	Yes | No (default)
Regex String	(Required) Specifies the Regex pattern to use in the search. **Note** This parameter is required unless Max Stream Length is set. Do not set the Regex String if Max Stream Length is set.	string
Service Ports	(Required) Specifies a comma-separated list of ports or port ranges where the target service resides. **Note** This parameter is required for the String XL TCP and String XL UDP signature engines. It cannot be used for the String XL ICMP signature engine.	0 to 65535[1] a-b[,c-d]

Table B-36 String XL Engine Parameters (continued) (continued)

Parameter	Description	Value
Specify Exact Match Offset {Yes | No}	Enables exact match offset: • Exact Match Offset—Specifies the exact stream offset in bytes the regular expression string must report for a match to be valid.	0 to 65535
Specify Maximum Match Offset {Yes | No}	Enables maximum match offset: • Maximum Match Offset—Specifies the maximum stream offset in bytes the regular expression string must report for a match to be valid.	0 to 65535
Specify Min Match Offset {Yes | No}	Enables minimum match offset: • Min Match Offset—Specifies the minimum stream offset in bytes the regular expression string must report for a match to be valid.	0 to 65535
Specify Max Match Length {Yes | No}	Enables maximum match length: • Max Match Length—Specifies the maximum number of bytes the regular expression string must match for the pattern to be considered a hit.	0 to 65535
Specify Min Match Length {Yes | No}	Enables minimum match length: • Min Match Length—Specifies the minimum number of bytes the regular expression string must match for the pattern to be considered a hit.	0 to 65535
Specify Max Stream Length {Yes | No}	Enables maximum stream length: • Max Stream Length—Limits the search to the first configured number of bytes. The length of the stream is checked again this value. If the stream contains more bytes than this value, an alert is triggered. **Note** When you specify this parameter, you cannot configure Raw Regex or Regex String.	Yes | No 0 to 65535
Specify Max Whole Length {Yes | No}	Enables maximum whole length: • Max Whole Length—Specifies the maximum length for the pattern that will not be fragmented.	Yes | No 0 to 65535
Specify Min Whole Length {Yes | No}	Enables minimum whole length: • Min Whole Length—Specifies the minimum length for the pattern that will not be fragmented.	Yes | No 0 to 65535

Table B-36 String XL Engine Parameters (continued) (continued)

Parameter	Description	Value
Stingy	Specifies to stop looking for larger matches after the first completed match. **Note** Stingy can only be used with Min Match Length; otherwise, it is ignored.	True | False (default)
Strip Telnet Options	Strips the Telnet option characters from the data before the pattern is searched.[2]	True | False (default)
Swap Attacker Victim	True if address (and ports) source and destination are swapped in the alert message. False for no swap (default).	Yes | No (default)
UTF8	Treats all legal UTF-8 byte sequences in the expression as a single character.	True | False (default)

[1] The second number in the range must be greater than or equal to the first number.

[2] This parameter is primarily used as an IPS anti-evasion tool.

Unsupported String XL Parameters

Although you see the End Optional and Specify Max Stream Length parameters in the String XL engine, they are disabled in IPS 7.1(*x*). and later., and IPS 7.2(*x*) and later. You receive an error message if you try to configure them. For example, here is the error message you receive after you create a signature using Specify Max Stream Length and then try to save it:

```
Apply Changes?[yes]: yes
Error: string-xl-tcp 60003.0 : Maximum Stream Length is currently not supported.
Please don't use this option.

The configuration changes failed validation, no changes were applied.
Would you like to return to edit mode to correct the errors? [yes]:
```

Sweep Engines

This section describes the Sweep engines, and contains the following topics:

- Sweep Engine, page B-68
- Sweep Other TCP Engine, page B-70

Sweep Engine

The Sweep engine analyzes traffic between two hosts or from one host to many hosts. You can tune the existing signatures or create custom signatures. The Sweep engine has protocol-specific parameters for ICMP, UDP, and TCP.

The alert conditions of the Sweep engine ultimately depend on the count of the unique parameter. The unique parameter is the threshold number of distinct hosts or ports depending on the type of sweep. The unique parameter triggers the alert when more than the unique number of ports or hosts is seen on the address set within the time period. The processing of unique port and host tracking is called counting.

Caution Event action filters based on source and destination IP addresses do not function for the Sweep engine, because they do not filter as regular signatures. To filter source and destination IP addresses in sweep alerts, use the source and destination IP address filter parameters in the Sweep engine signatures.

A unique parameter must be specified for all signatures in the Sweep engine. A limit of 2 through 40 (inclusive) is enforced on the sweeps. 2 is the absolute minimum for a sweep, otherwise, it is not a sweep (of one host or port). 40 is a practical maximum that must be enforced so that the sweep does not consume excess memory. More realistic values for unique range between 5 and 15.

TCP sweeps must have a TCP flag and mask specified to determine which sweep inspector slot in which to count the distinct connections. ICMP sweeps must have an ICMP type specified to discriminate among the various types of ICMP packets.

Data Node

When an activity related to Sweep engine signatures is seen, the IPS uses a data node to determine when it should stop monitoring for a particular host. The data node contains various persistent counters and variables needed for cross-packet reassembly of streams and for tracking the inspection state on a per-stream/per-source/per-destination basis The data node containing the sweep determines when the sweep should expire. The data node stops a sweep when the data node has not seen any traffic for *x* number of seconds (depending on the protocol).

There are several adaptive timeouts for the data nodes. The data node expires after 30 seconds of idle time on the address set after all of the contained objects have been removed. Each contained object has various timeouts, for example, TCP Stream has a one-hour timeout for established connections. Most other objects have a much shorter expiration time, such as 5 or 60 seconds.

Table B-37 lists the parameters specific to the Sweep engine.

Table B-37 Sweep Engine Parameters

Parameter	Description	Value
Destination Address Filter	Specifies the destination IP address to exclude from the sweep counting algorithm.	<A.B.C.D>-<A.B.C.D> [,<A.B.C.D>-<A.B.C.D>]
Source Address Filter	Specifies the source IP address to exclude from the sweep counting algorithm.	<A.B.C.D>-<A.B.C.D> [,<A.B.C.D>-<A.B.C.D>]
Protocol	Specifies the protocol of interest for this inspector.	ICMP UDP TCP
Specify ICMP Type {Yes | No}	(Optional) Enables the ICMP header type: • ICMP Type—Specifies the value of the ICMP header TYPE.	0 to 255
Specify Port Range {Yes | No}	(Optional) Enables using a port range for inspection: • Port Range—Specifies the UDP port range used in inspection.	0 to 65535 a-b[,c-d]

Table B-37 Sweep Engine Parameters (continued)

Parameter	Description	Value
Fragment Status	Specifies whether fragments are wanted or not: • Any fragment status • Do not inspect fragments • Inspect fragments	Any No Fragment Want Fragment
Inverted Sweep	Uses source port instead of destination port for unique counting.	True | False
Mask	Specifies the mask used in TCP flags comparison: • URG bit • ACK bit • PSH bit • RST bit • SYN bit • FIN bit	URG ACK PSH RST SYN FIN
Storage Key	Specifies the type of address key used to store persistent data: • Attacker address • Attacker and victim addresses • Attacker address and victim port	Axxx AxBx Axxb
Suppress Reverse	Does not fire when a sweep has fired in the reverse direction on this address set.	Yes | No
Swap Attacker Victim	Swaps the attacker and victim addresses and ports (source and destination) in the alert message and in any actions taken.	Yes | No (default)
TCP Flags	Specifies the TCP flags to match when masked by mask: • URG bit • ACK bit • PSH bit • RST bit • SYN bit • FIN bit	URG ACK PSH RST SYN FIN
Unique	Specifies the threshold number of unique port connections between the two hosts.	0 to 65535

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Sweep Other TCP Engine

The Sweep Other TCP engine analyzes traffic between two hosts looking for abnormal packets typically used to fingerprint a victim. You can tune the existing signatures or create custom signatures.

TCP sweeps must have a TCP flag and mask specified. You can specify multiple entries in the set of TCP flags. And you can specify an optional port range to filter out certain packets.

Table B-38 lists the parameters specific to the Sweep Other TCP engine.

Table B-38 Sweep Other TCP Engine Parameters

Parameter	Description	Value
Specify Port Range {Yes | No}	(Optional) Enables using a port range for inspection: • Port Range—Specifies the UDP port range used in inspection.	0 to 65535 a-b[,c-d]
Set TCP Flags	Lets you set TCP flags to match. • TCP Flags—Specifies the TCP flags used in this inspection: – URG bit – ACK bit – PSH bit – RST bit – SYN bit – FIN bit	URG ACK PSH RST SYN FIN

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Traffic Anomaly Engine

Note You can edit or tune anomaly detection signatures but you cannot create custom anomaly detection signatures.

The Traffic Anomaly engine contains nine anomaly detection signatures covering the three protocols (TCP, UDP, and other). Each signature has two subsignatures, one for the scanner and the other for the worm-infected host (or a scanner under worm attack). When anomaly detection discovers an anomaly, it triggers an alert for these signatures. All anomaly detection signatures are enabled by default and the alert severity for each one is set to high.

When a scanner is detected but no histogram anomaly occurred, the scanner signature fires for that attacker (scanner) IP address. If the histogram signature is triggered, the attacker addresses that are doing the scanning each trigger the worm signature (instead of the scanner signature). The alert details state which threshold is being used for the worm detection now that the histogram has been triggered. From that point on, all scanners are detected as worm-infected hosts.

The following anomaly detection event actions are possible:

- Product Alert—Writes the event to the Event Store.
- Deny Attacker Inline—Does not transmit this packet and future packets originating from the attacker address for a specified period of time.

- Log Attacker Packets—Starts IP logging for packets that contain the attacker address.
- Log Attacker/Victim Pair Packets—Starts IP logging for packets that contain the attacker and victim address pair.

Note For deny actions, to set the specified period of time and maximum number of denied attackers, choose **Configuration > Policies > Event Action Rules > rules0 > General Settings**.

- Deny Attacker Service Pair Inline—Blocks the source IP address and the destination port.
- Request SNMP Trap—Sends a request to NotificationApp to perform SNMP notification.
- Request Block Host—Sends a request to ARC to block this host (the attacker).

Table B-39 lists the anomaly detection worm signatures.

Table B-39 Anomaly Detection Worm Signatures

Signature ID	Subsignature ID	Name	Description
13000	0	Internal TCP Scanner	Identified a single scanner over a TCP protocol in the internal zone.
13000	1	Internal TCP Scanner	Identified a worm attack over a TCP protocol in the internal zone; the TCP histogram threshold was crossed and a scanner over a TCP protocol was identified.
13001	0	Internal UDP Scanner	Identified a single scanner over a UDP protocol in the internal zone.
13001	1	Internal UDP Scanner	Identified a worm attack over a UDP protocol in the internal zone; the UDP histogram threshold was crossed and a scanner over a UDP protocol was identified.
13002	0	Internal Other Scanner	Identified a single scanner over an Other protocol in the internal zone.
13002	1	Internal Other Scanner	Identified a worm attack over an Other protocol in the internal zone; the Other histogram threshold was crossed and a scanner over an Other protocol was identified.
13003	0	External TCP Scanner	Identified a single scanner over a TCP protocol in the external zone.
13003	1	External TCP Scanner	Identified a worm attack over a TCP protocol in the external zone; the TCP histogram threshold was crossed and a scanner over a TCP protocol was identified.
13004	0	External UDP Scanner	Identified a single scanner over a UDP protocol in the external zone.

Table B-39 Anomaly Detection Worm Signatures (continued)

Signature ID	Subsignature ID	Name	Description
13004	1	External UDP Scanner	Identified a worm attack over a UDP protocol in the external zone; the UDP histogram threshold was crossed and a scanner over a UDP protocol was identified.
13005	0	External Other Scanner	Identified a single scanner over an Other protocol in the external zone.
13005	1	External Other Scanner	Identified a worm attack over an Other protocol in the external zone; the Other histogram threshold was crossed and a scanner over an Other protocol was identified.
13006	0	Illegal TCP Scanner	Identified a single scanner over a TCP protocol in the illegal zone.
13006	1	Illegal TCP Scanner	Identified a worm attack over a TCP protocol in the illegal zone; the TCP histogram threshold was crossed and a scanner over a TCP protocol was identified.
13007	0	Illegal UDP Scanner	Identified a single scanner over a UDP protocol in the illegal zone.
13007	1	Illegal UDP Scanner	Identified a worm attack over a UDP protocol in the illegal zone; the UDP histogram threshold was crossed and a scanner over a UDP protocol was identified.
13008	0	Illegal Other Scanner	Identified a single scanner over an Other protocol in the illegal zone.
13008	1	Illegal Other Scanner	Identified a worm attack over an Other protocol in the illegal zone; the Other histogram threshold was crossed and a scanner over an Other protocol was identified.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Traffic ICMP Engine

The Traffic ICMP engine analyzes nonstandard protocols, such as TFN2K, LOKI, and DDoS. There are only two signatures (based on the LOKI protocol) with user-configurable parameters.

TFN2K is the newer version of the TFN. It is a DDoS agent that is used to control coordinated attacks by infected computers (zombies) to target a single computer (or domain) with bogus traffic floods from hundreds or thousands of unknown attacking hosts. TFN2K sends randomized packet header information, but it has two discriminators that can be used to define signatures. One is whether the L3 checksum is incorrect and the other is whether the character 64 ‘A’ is found at the end of the payload. TFN2K can run on any port and can communicate with ICMP, TCP, UDP, or a combination of these protocols.

LOKI is a type of back door Trojan. When the computer is infected, the malicious code creates an ICMP Tunnel that can be used to send small payload in ICMP replies (which may go straight through a firewall if it is not configured to block ICMP.) The LOKI signatures look for an imbalance of ICMP echo requests to replies and simple ICMP code and payload discriminators.

The DDoS category (excluding TFN2K) targets ICMP-based DDoS agents. The main tools used here are TFN and Stacheldraht. They are similar in operation to TFN2K, but rely on ICMP only and have fixed commands: integers and strings.

Table B-40 lists the parameters specific to the Traffic ICMP engine.

Table B-40 Traffic ICMP Engine Parameters

Parameter	Description	Value
Parameter Tunable Sig	Specifies the whether this signature has configurable parameters.	Yes | No
Inspection Type	Specifies the type of inspection to perform: • Inspects for original LOKI traffic • Inspects for modified LOKI traffic	is Loki Is Mod Loki
Reply Ratio	Specifies the imbalance of replies to requests. The alert fires when there are this many more replies than requests.	0 to 65535
Want Request	Requires an ECHO REQUEST be seen before firing the alert.	True | False

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.

Trojan Engines

The Trojan engines analyze nonstandard protocols, such as BO2K and TFN2K. There are three Trojan engines: Trojan BO2K, TrojanTFN2K, and Trojan UDP.

BO was the original Windows back door Trojan that ran over UDP only. It was soon superseded by BO2K. BO2K supported UDP and TCP both with basic XOR encryption. They have plain BO headers that have certain cross-packet characteristics.

BO2K also has a stealthy TCP module that was designed to encrypt the BO header and make the cross-packet patterns nearly unrecognizable. The UDP modes of BO and BO2K are handled by the Trojan UDP engine. The TCP modes are handled by the Trojan BO2K engine.

Note There are no specific parameters to the Trojan engines, except for Swap Attacker Victim in the Trojan UDP engine.

For More Information

For more information on the parameters common to all signature engines, see Master Engine, page B-4.