# SIP Inspector

## SIP Inspector Overview

| | |
|---|---|
| Type | Inspector (service) |
| Usage | Inspect |
| Instance Type | Multiton |
| Other Inspectors Required | `stream_udp` |
| Enabled | `true` |

The Session Initiation Protocol (SIP) manages the creation, modification, and teardown of real-time call sessions that include one or more participants. The applications that SIP can control include: internet telephony, multimedia conferencing, instant messaging, online gaming, and file transfer. The SIP protocol is a text-based, request and response protocol.

A SIP request includes a `method` field that identifies the purpose of the request, and a `Request-URI` which specifies where to send the request. A status code in each SIP response indicates the outcome of the requested action. The SIP protocol uses TCP (port 5060) or UDP (port 5061).

After SIP creates a call session, SIP can transmit audio and video streams over the real-time transport protocol (RTP). The SIP message body embeds the data-channel parameter negotiation, session announcement, and session invitation in the Session Description Protocol (SDP) format.

The `sip` inspector detects and analyzes SIP messages in network traffic. The `sip` inspector extracts the SIP header and message body and passes any data in the SIP message body to the detection engine.

The `sip` inspector detects anomalies and known vulnerabilities in SIP traffic, including disordered and invalid call sequences.

**Note**

- The sip inspector does not decode RTP messages. The sip inspector identifies the RTP channel based on the port defined in the SDP data.
- UDP typically carries media sessions supported by SIP. The sip inspector obtains session tracking information from the decoded UDP stream.
- SIP rule options allow you to position the detection cursor to the SIP packet header or message body and to limit detection to packets for specific SIP methods or status codes.

# SIP Inspector Parameters

### SIP service configuration

The binder inspector defines the SIP service configuration. For more information, see the Binder Inspector Overview.

**Example:**

```
[
    {
        "when": {
            "role": "any",
            "service": "sip"
        },
        "use": {
            "type": "sip"
        }
    }
]
```

### ignore_call_channel

Specifies whether to inspect audio/video data channel traffic. When enabled, the sip inspector decodes all non-data SIP channel traffic and ignores audio/video SIP data channel traffic.

**Type:** boolean

**Valid values:** true, false

**Default value:** false

### max_call_id_len

Specifies the maximum number of bytes to allow in the Call-ID header field. The Call-ID field uniquely identifies the SIP session in requests and responses. The sip inspector does not generate an alert when the max_call_id_len is 0.

You can enable rule 140:5 to generate events, and in an inline deployment, drop offending packets. The sip inspector generates an event when the Call-ID header length is greater than the value of max_call_id_len.

**Type:** integer

**Valid range:** 0 to 65535

**Default value:** 256

### max_contact_len

Specifies the maximum number of bytes to allow in the `Contact` header field. The `Contact` field provides a URI that specifies the location to contact with subsequent messages. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:15 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the `Contact` header field length is greater than the value of `max_contact_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `256`

### max_content_len

Specifies the maximum number of bytes to allow in the content of the message body. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:16 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the content length is greater than the value of `max_content_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `1024`

### max_dialogs

Specifies the maximum number of dialogs allowed within a stream session. If the number of dialogs is more than the set limit, the `sip` inspector drops the oldest dialogs until the number of dialogs does not exceed the maximum number specified.

You can enable rule 140:27 to generate events, and in an inline deployment, drop offending packets.

**Type:** integer

**Valid range:** `1` to `4,294,967,295 (max32)`

**Default value:** `4`

### max_from_len

Specifies the maximum number of bytes to allow in the `From` header field. The `From` field identifies the sender of the message. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:9 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the `From` field length is greater than the value of `max_from_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `256`

### max_request_name_len

Specifies the maximum number of bytes to allow in the request name. The SIP request name refers to the name of the method specified in the SIP `CSeq` transaction identifier. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:7 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the request name length is greater than the value of `max_request_name_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `20`

### max_requestName_len

The `max_requestName_len` parameter is deprecated. Use the `max_request_name_len` parameter instead.

### max_to_len

Specifies the maximum number of bytes to allow in the `To` header field. The `To` field identifies the recipient of the message. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:11 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the `To` field length is greater than the value of `max_to_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `256`

### max_uri_len

Specifies the maximum number of bytes to allow in the SIP `Request-URI`. The `Request-URI` indicates the destination path to the requested resource. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:3 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the `Request-URI` field length is greater than the value of `max_uri_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `256`

### max_via_len

Specifies the maximum number of bytes to allow in the `Via` header field. The `Via` field identifies the transport to use in the request and the location of the recipient. The `sip` inspector does not generate an alert when the value is 0.

You can enable rule 140:13 to generate events, and in an inline deployment, drop offending packets. The `sip` inspector generates an event when the `Via` field length is greater than the value of `max_via_len`.

**Type:** integer

**Valid range:** `0` to `65535`

**Default value:** `1024`

### methods

Specifies a list of SIP methods to detect. Method names are case-insensitive. Use a comma or space to separate method names in the list. A method name can only include alphabetic characters, numbers, and the underscore character.

**Type:** string

**Valid values:** `ack`, `benotify`, `bye`, `cancel`, `do`, `info`, `invite`, `join`, `message`, `notify`, `options`, `prack`, `publish`, `quath`, `refer`, `register`, `service`, `sprack`, `subscribe`, `unsubscribe`, `update`

**Default value:** `invite cancel ack bye register options`

# SIP Inspector Rules

Enable the `sip` inspector rules to generate events and, in an inline deployment, drop offending packets.

*Table 1: SIP Inspector Rules*

| GID:SID | Rule Message |
|---|---|
| 140:2 | empty request URI |
| 140:3 | URI is too long |
| 140:4 | empty call-Id |
| 140:5 | Call-Id is too long |
| 140:6 | CSeq number is too large or negative |
| 140:7 | request name in CSeq is too long |
| 140:8 | empty From header |
| 140:9 | From header is too long |
| 140:10 | empty To header |
| 140:11 | To header is too long |
| 140:12 | empty Via header |
| 140:13 | Via header is too long |
| 140:14 | empty Contact |
| 140:15 | contact is too long |
| 140:16 | content length is too large or negative |
| 140:17 | multiple SIP messages in a packet |
| 140:18 | content length mismatch |
| 140:19 | request name is invalid |

| GID:SID | Rule Message |
|---|---|
| 140:20 | Invite replay attack |
| 140:21 | illegal session information modification |
| 140:22 | response status code is not a 3 digit number |
| 140:23 | empty Content-type header |
| 140:24 | SIP version is invalid |
| 140:25 | mismatch in METHOD of request and the CSEQ header |
| 140:26 | method is unknown |
| 140:27 | maximum dialogs within a session reached |

# SIP Inspector Intrusion Rule Options

### sip_method

A SIP request method identifies the purpose of the request. Use the `sip_method` keyword to match the method in a SIP request. Method names are case-insensitive. Separate multiple method names with a comma.

**Type:** string

**Syntax:** `sip_method: <methods>;`

**Valid values:** `ack`, `benotify`, `bye`, `cancel`, `do`, `info`, `invite`, `join`, `message`, `notify`, `options`, `prack`, `publish`, `quath`, `refer`, `register`, `service`, `sprack`, `subscribe`, `unsubscribe`, `update`

**Examples:** `sip_method: "ack,service,info,bye";`

### sip_stat_code

A SIP response includes a three-digit status code. The SIP status code indicates the outcome of the requested action. Use the `sip_stat_code` keyword to match a SIP response with the specified status codes.

You can specify a one-digit number that represents the first digit of a three-digit status code, a three-digit number, or a comma-separated list of numbers using either number combination. A list matches if any single number in the list matches the code in the SIP response.

**Type:** integer

**Syntax:** `sip_stat_code: <codes>;`

**Valid ranges:**

- `1` to `9`
- `100` to `999`

**Examples:** `sip_stat_code: "1";`

*Table 2: SIP Parameter Values and Status Codes*

| Parameter Value | Detected Status Codes | Description |
|---|---|---|
| 189 | 189 | Set a specific status code. |
| 1 | 100 - 199 | Set a single digit. |
| 222, 3 | 222; 300 - 399 | Set a comma-separated list of three-digit or single digit numbers. |

### sip_header

Use the `sip_header` keyword to position the detection cursor to the beginning of the extracted SIP header buffer. Restricts inspection to the header fields.

**Syntax:** `sip_header;`

**Examples:** `sip_header;`

### sip_body

Use the `sip_body` keyword to position the detection cursor to the beginning of the extracted SIP message body. Restricts inspection to the message body.

**Syntax:** `sip_body;`

**Examples:** `sip_body;`

**Note** The `sip` inspector extracts the entire message body and makes it available to the rules engine. The rules engine is not limited to searching for session description protocol (SDP) content.