Stream:	Internet Engineering Task Force (IETF)				
RFC:	8936				
Category:	Standards Track				
Published:	November 2020				
ISSN:	2070-1721				
Authors:	A. Backman, Ed. *Amazon*	M. Jones, Ed. *Microsoft*	M. Scurtescu *Coinbase*	M. Ansari *Independent*	A. Nadalin *Independent*

RFC 8936
Poll-Based Security Event Token (SET) Delivery Using HTTP

Abstract

This specification defines how a series of Security Event Tokens (SETs) can be delivered to an intended recipient using HTTP POST over TLS initiated as a poll by the recipient. The specification also defines how delivery can be assured, subject to the SET Recipient's need for assurance.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and Overview

This specification defines how a stream of Security Event Tokens (SETs) [RFC8417] can be transmitted to an intended SET Recipient using HTTP [RFC7231] over TLS. The specification defines a method to poll for SETs using HTTP POST. This is an alternative SET delivery method to the one defined in [RFC8935].

Poll-based SET delivery is intended for scenarios where all of the following apply:

- The recipient of the SET is capable of making outbound HTTP requests.
- The transmitter is capable of hosting a TLS-enabled HTTP endpoint that is accessible to the recipient.
- The transmitter and recipient are willing to exchange data with one another.

In some scenarios, either push-based or poll-based delivery could be used, and in others, only one of them would be applicable.

A mechanism for exchanging configuration metadata such as endpoint URLs, cryptographic keys, and possible implementation constraints such as buffer size limitations between the transmitter and recipient is out of scope for this specification. How SETs are defined and the process by which security events are identified for SET Recipients are specified in [RFC8417].

1.1. Notational Conventions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Throughout this document, all figures may contain spaces and extra line wrapping for readability and due to space limitations.

1.2. Definitions

This specification utilizes terminology defined in [RFC8417] and [RFC8935].

2. SET Delivery

When a SET is available for a SET Recipient, the SET Transmitter queues the SET in a buffer so that a SET Recipient can poll for SETs using HTTP POST.

In poll-based SET delivery using HTTP over TLS, zero or more SETs are delivered in a JSON [RFC8259] document to a SET Recipient in response to an HTTP POST request to the SET Transmitter. Then in a following request, the SET Recipient acknowledges received SETs and can poll for more. All requests and responses are JSON documents and use a `Content-Type` of `application/json`, as described in Section 2.2.

After successful (acknowledged) SET delivery, SET Transmitters are not required to retain or record SETs for retransmission. Once a SET is acknowledged, the SET Recipient **SHALL** be responsible for retention, if needed. Transmitters may also discard undelivered SETs under deployment-specific conditions, such as if they have not been polled for over too long a period of time or if an excessive amount of storage is needed to retain them.

Upon receiving a SET, the SET Recipient reads the SET and validates it in the manner described in Section 2 of [RFC8935]. The SET Recipient **MUST** acknowledge receipt to the SET Transmitter, and **SHOULD** do so in a timely fashion, as described in Section 2.4. The SET Recipient **SHALL NOT** use the event acknowledgement mechanism to report event errors other than those relating to the parsing and validation of the SET.

2.1. Polling Delivery using HTTP

This method allows a SET Recipient to use HTTP POST (Section 4.3.3 of [RFC7231]) to acknowledge SETs and to check for and receive zero or more SETs. Requests **MAY** be made at a periodic interval (short polling) or requests **MAY** wait, pending availability of new SETs using long polling, per Section 2 of [RFC6202]. Note that short polling will result in retrieving zero or more SETs whereas long polling will typically result in retrieving one or more SETs unless a timeout occurs.

The delivery of SETs in this method is facilitated by HTTP POST requests initiated by the SET Recipient in which:

- The SET Recipient makes a request for available SETs using an HTTP POST to a pre-arranged endpoint provided by the SET Transmitter, or
- after validating previously received SETs, the SET Recipient initiates another poll request using HTTP POST that includes acknowledgement of previous SETs and requests the next batch of SETs.

The purpose of the acknowledgement is to inform the SET Transmitter that delivery has succeeded and redelivery is no longer required. Before acknowledgement, SET Recipients validate the received SETs and retain them in a manner appropriate to the recipient's requirements. The level and method of retention of SETs by SET Recipients is out of scope of this specification.

2.2. Polling HTTP Request

When initiating a poll request, the SET Recipient constructs a JSON document that consists of polling request parameters and SET acknowledgement parameters in the form of JSON objects.

When making a request, the HTTP `Content-Type` header field is set to `application/json`.

The following JSON object members are used in a polling request:

Request Processing Parameters

maxEvents

An **OPTIONAL** integer value indicating the maximum number of unacknowledged SETs to be returned. The SET Transmitter **SHOULD NOT** send more SETs than the specified maximum. If more than the maximum number of SETs are available, the SET Transmitter determines which to return first; the oldest SETs available **MAY** be returned first, or another selection algorithm **MAY** be used, such as prioritizing SETs in some manner that makes sense for the use case. A value of `0` **MAY** be used by SET Recipients that would like to perform an acknowledge-only request. This enables the Recipient to use separate HTTP requests for acknowledgement and reception of SETs. If this parameter is omitted, no limit is placed on the number of SETs to be returned.

returnImmediately

An **OPTIONAL** JSON boolean value that indicates the SET Transmitter **SHOULD** return an immediate response even if no results are available (short polling). The default value is `false`, which indicates the request is to be treated as an HTTP long poll, per Section 2 of [RFC6202]. The timeout for the request is part of the configuration between the participants, which is out of scope of this specification.

SET Acknowledgment Parameters

ack

A JSON array of strings whose values are the `jti` [RFC7519] values of successfully received SETs that are being acknowledged. If there are no outstanding SETs to acknowledge, this member is omitted or contains an empty array. Once a SET has been acknowledged, the SET Transmitter is released from any obligation to retain the SET.

setErrs

A JSON object with one or more members whose keys are the `jti` values of invalid SETs received. The values of these objects are themselves JSON objects that describe the errors detected using the `err` and `description` values specified in Section 2.6. If there are no outstanding SETs with errors to report, this member is omitted or contains an empty JSON object.

2.3. Polling HTTP Response

In response to a poll request, the SET Transmitter checks for available SETs and responds with a JSON document containing the following JSON object members:

sets

A JSON object containing zero or more SETs being returned. Each member name is the `jti` of a SET to be delivered, and its value is a JSON string representing the corresponding SET. If there are no outstanding SETs to be transmitted, the JSON object **SHALL** be empty. Note that both SETs being transmitted for the first time and SETs that are being retransmitted after not having been acknowledged are communicated here.

moreAvailable

A JSON boolean value that indicates if more unacknowledged SETs are available to be returned. This member **MAY** be omitted, with the meaning being the same as including it with the boolean value `false`.

When making a response, the HTTP `Content-Type` header field is set to `application/json`.

2.4. Poll Request

The SET Recipient performs an HTTP POST (see Section 4.3.4 of [RFC7231]) to a pre-arranged polling endpoint URI to check for SETs that are available. Because the SET Recipient has no prior SETs to acknowledge, the `ack` and `setErrs` request parameters are omitted.

After a period of time configured in an out-of-band manner between the SET Transmitter and Recipient, a SET Transmitter **MAY** redeliver SETs it has previously delivered. The SET Recipient **SHOULD** accept repeat SETs and acknowledge the SETs regardless of whether the Recipient believes it has already acknowledged the SETs previously. A SET Transmitter **MAY** limit the number of times it attempts to deliver a SET.

If the SET Recipient has received SETs from the SET Transmitter, the SET Recipient parses and validates that received SETs meet its own requirements and **SHOULD** acknowledge receipt in a timely fashion (e.g., seconds or minutes) so that the SET Transmitter can mark the SETs as received. SET Recipients **SHOULD** acknowledge receipt before taking any local actions based on the SETs to avoid unnecessary delay in acknowledgement, where possible.

Poll requests have three variations:

Poll-Only

In this scenario, a SET Recipient asks for the next set of events where no previous SET deliveries are acknowledged (such as in the initial poll request).

Acknowledge-Only

In this scenario, a SET Recipient sets the `maxEvents` value to `0` along with `ack` and `setErrs` members indicating the SET Recipient is acknowledging previously received SETs and does not want to receive any new SETs in response to the request.

Combined Acknowledge and Poll
: In this scenario, a SET Recipient is both acknowledging previously received SETs using the `ack` and `setErrs` members and will wait for the next group of SETs in the SET Transmitters response.

2.4.1. Poll-Only Request

In the case where no SETs were received in a previous poll (see Figure 7), the SET Recipient simply polls without acknowledgement parameters (`ack` and `setErrs`).

The following is a non-normative example request made by a SET Recipient that has no outstanding SETs to acknowledge and is polling for available SETs at the endpoint `https://notify.idp.example.com/Events`:

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{
 "returnImmediately": true
}
```

Figure 1: Example Initial Poll Request

A SET Recipient can poll using default parameter values by passing an empty JSON object.

The following is a non-normative example default poll request to the endpoint `https://notify.idp.example.com/Events`:

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{}
```

Figure 2: Example Default Poll Request

2.4.2. Acknowledge-Only Request

In this variation, the SET Recipient acknowledges previously received SETs and indicates it does not want to receive SETs in response by setting the `maxEvents` value to `0`. This variation might be used, for instance, when a SET Recipient needs to acknowledge received SETs independently (e.g., on separate threads) from the process of receiving SETs.

If the poll needs to return immediately, then `returnImmediately` **MUST** also be present with the value `true`. If it is `false`, then a long poll will still occur until an event is ready to be returned, even though no events will be returned.

The following is a non-normative example poll request with acknowledgement of SETs received (for example, as shown in Figure 6):

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{
  "ack": [
    "4d3559ec67504aaba65d40b0363faad8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ],
  "maxEvents": 0,
  "returnImmediately": true
}
```

Figure 3: Example Acknowledge-Only Request

2.4.3. Poll with Acknowledgement

This variation allows a recipient thread to simultaneously acknowledge previously received SETs and wait for the next group of SETs in a single request.

The following is a non-normative example poll with acknowledgement of the SETs received in Figure 6:

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{
  "ack": [
    "4d3559ec67504aaba65d40b0363faad8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ],
  "returnImmediately": false
}
```

Figure 4: Example Poll with Acknowledgement and No Errors

In the above acknowledgement, the SET Recipient has acknowledged receipt of two SETs and has indicated it wants to wait until the next SET is available.

2.4.4. Poll with Acknowledgement and Errors

In the case where errors were detected in previously delivered SETs, the SET Recipient **MAY** use the `setErrs` member to communicate the errors in the following poll request.

The following is a non-normative example of a response acknowledging one successfully received SET and one SET with an error from the two SETs received in Figure 6:

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Language: en-US
Content-Type: application/json

{
  "ack": ["3d0c3cf797584bd193bd0fb1bd4e7d30"],
  "setErrs": {
    "4d3559ec67504aaba65d40b0363faad8": {
      "err": "authentication_failed",
      "description": "The SET could not be authenticated"
    }
  },
  "returnImmediately": true
}
```

Figure 5: Example Poll Acknowledgement with Error

2.5. Poll Response

In response to a valid poll request, the service provider **MAY** respond immediately if SETs are available to be delivered. If no SETs are available at the time of the request, the SET Transmitter **SHALL** delay responding until a SET is available or the timeout interval has elapsed unless the poll request parameter `returnImmediately` is present with the value `true`.

As described in Section 2.3, a JSON document is returned containing members including `sets`, which **SHALL** contain zero or more SETs.

The following is a non-normative example response to the request shown in Section 2.4. This example shows two SETs being returned:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "sets":
  {
   "4d3559ec67504aaba65d40b0363faad8":
   "eyJhbGciOiJub25lIn0.
    eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFkOCIsImlhdC
    I6MTQ1ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwi
    YXVkIjpbImh0dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MW
    ZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0Zl
    ZWRzLzVkNzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybj
    ppZXRmOnBhcmFtczpzY2ltOmV2ZW50OmNyZWF0ZSI6eyJyZWYiOiJodHRwczov
    L3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTc1Mj
    FkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuYW1lIiwidXNlck5hbWUiLCJwYXNz
    d29yZCIsImVtYWlscyJdfX19.",
   "3d0c3cf797584bd193bd0fb1bd4e7d30":
   "eyJhbGciOiJub25lIn0.
    eyJqdGkiOiIzZDBjM2NmNzk3NTg0YmQxOTNiZDBmYjFiZDRlN2QzMCIsImlhdC
    I6MTQ1ODQ5NjAyNSwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwi
    YXVkIjpbImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MW
    ZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9qaHViLmV4YW1wbGUuY29tL0Zl
    ZWRzLzVkNzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwic3ViIjoiaHR0cHM6Ly
    9zY2ltLmV4YW1wbGUuY29tL1VzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIx
    ZDkiLCJldmVudHMiOnsidXJuOmlldGY6cGFyYW1zOnNjaW06ZXZlbnQ6cGFzc3
    dvcmRSZXNldCI6eyJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkifSwi
    aHR0cHM6Ly9leGFtcGxlLmNvbS9zY2ltL2V2ZW50L3Bhc3N3b3JkUmVzZXRFeH
    QiOnsicmVzZXRBdHRlbXB0cyI6NX19fQ."
  }
}
```

Figure 6: Example Poll Response

In the above example, two SETs whose `jti` values are `4d3559ec67504aaba65d40b0363faad8` and `3d0c3cf797584bd193bd0fb1bd4e7d30` are delivered.

The following is a non-normative example response to the request shown in Section 2.4.1, which indicates that no new SETs or unacknowledged SETs are available:

```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
   "sets": {}
  }
```

Figure 7: Example No SETs Poll Response

Upon receiving the JSON document (e.g., as shown in Figure 6), the SET Recipient parses and verifies the received SETs and notifies the SET Transmitter of successfully received SETs and SETs with errors via the next poll request to the SET Transmitter, as described in Sections 2.4.3 and 2.4.4.

2.5.1. Poll Error Response

In the event of a general HTTP error condition in the context of processing a poll request, the service provider responds with the applicable HTTP response status code, as defined in Section 6 of [RFC7231].

Service providers **MAY** respond to any invalid poll request with an HTTP response status code of 400 (Bad Request) even when a more specific code might apply, for example, if the service provider deemed that a more specific code presented an information disclosure risk. When no more specific code might apply, the service provider **SHALL** respond to an invalid poll request with an HTTP status code of 400.

The response body for responses to invalid poll requests is left undefined, and its contents **SHOULD** be ignored.

The following is a non-normative example of a response to an invalid poll request:

```
HTTP/1.1 400 Bad Request
```

Figure 8: Example Poll Error Response

2.6. Error Response Handling

If a SET is invalid, error codes from the IANA "Security Event Token Error Codes" registry established by [RFC8935] are used in error responses. As described in Section 2.3 of [RFC8935], an error response is a JSON object providing details about the error that includes the following name/value pairs:

err: A value from the IANA "Security Event Token Error Codes" registry that identifies the error.

description: A human-readable string that provides additional diagnostic information.

When included as part of a batch of SETs, the above JSON is included as part of the `setErrs` member, as defined in Sections 2.2 and 2.4.4.

When the SET Recipient includes one or more error responses in a request to the SET Transmitter, it must also include in the request a `Content-Language` header field whose value indicates the language of the error descriptions included in the request. The method of language selection in the case when the SET Recipient can provide error messages in multiple languages is out of scope for this specification.

3. Authentication and Authorization

The SET delivery method described in this specification is based upon HTTP over TLS [RFC2818] and standard HTTP authentication and authorization schemes, as per [RFC7235]. The TLS server certificate **MUST** be validated using DNS-ID [RFC6125] and/or DNS-Based Authentication of Named Entities (DANE) [RFC6698]. As per Section 4.1 of [RFC7235], a SET delivery endpoint **SHALL** indicate supported HTTP authentication schemes via the `WWW-Authenticate` header field when using HTTP authentication.

Authorization for the eligibility to provide actionable SETs can be determined by using the identity of the SET Issuer, validating the identity of the SET Transmitter, or via other employed authentication methods. Likewise, the SET Transmitter may choose to validate the identity of the SET Recipient, perhaps using mutual TLS. Because SETs are not commands, SET Recipients are free to ignore SETs that are not of interest after acknowledging their receipt.

4. Security Considerations

4.1. Authentication Using Signed SETs

JWS signed SETs can be used (see [RFC7515] and Section 5 of [RFC8417]) to enable the SET Recipient to validate that the SET Issuer is authorized to provide actionable SETs.

4.2. HTTP Considerations

SET delivery depends on the use of the Hypertext Transfer Protocol and is thus subject to the security considerations of HTTP (Section 9 of [RFC7230]) and its related specifications.

4.3. Confidentiality of SETs

SETs may contain sensitive information, including Personally Identifiable Information (PII), or be distributed through third parties. In such cases, SET Transmitters and SET Recipients **MUST** protect the confidentiality of the SET contents. In some use cases, using TLS to secure the transmitted SETs will be sufficient. In other use cases, encrypting the SET as described in JSON Web Encryption (JWE) [RFC7516] will also be required. The Event delivery endpoint **MUST** support at least TLS version 1.2 [RFC5246] and **SHOULD** support the newest version of TLS that meets its security requirements, which as of the time of this publication is TLS 1.3 [RFC8446]. The client **MUST** perform a TLS/SSL server certificate check using DNS-ID [RFC6125] and/or DANE [RFC6698]. How a SET Recipient determines the expected service identity to match the SET Transmitter's server certificate against is out of scope for this document. The implementation security considerations for TLS in "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)" [RFC7525] **MUST** be followed.

4.4. Access Token Considerations

If HTTP Authentication is performed using OAuth access tokens [RFC6749], implementers **MUST** take into account the threats and countermeasures documented in Section 8 of [RFC7521].

4.4.1. Bearer Token Considerations

Transmitting bearer tokens [RFC6750] using TLS helps prevent their interception.

Bearer tokens **SHOULD** have a limited lifetime that can be determined directly or indirectly (e.g., by checking with a validation service) by the service provider. By expiring tokens, clients are forced to obtain a new token (which usually involves re-authentication) for continued authorized access. For example, in OAuth 2.0, a client **MAY** use an OAuth refresh token to obtain a new bearer token after authenticating to an authorization server, per Section 6 of [RFC6749].

Implementations supporting OAuth bearer tokens need to factor in security considerations of this authorization method [RFC7521]. Since security is only as good as the weakest link, implementers also need to consider authentication choices coupled with OAuth bearer tokens. The security considerations of the default authentication method for OAuth bearer tokens, HTTP Basic, are well documented in [RFC7617]; therefore, implementers are encouraged to prefer stronger authentication methods.

5. Privacy Considerations

SET Transmitters should attempt to deliver SETs that are targeted to the specific business and protocol needs of subscribers.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, SET Transmitters and Recipients **MUST** have the appropriate legal agreements and user consent or terms of service in place. Furthermore, data that needs confidentiality protection **MUST** be encrypted, at least with TLS and sometimes also using JSON Web Encryption (JWE) [RFC7516].

In some cases, subject identifiers themselves may be considered sensitive information, such that their inclusion within a SET may be considered a violation of privacy. SET Issuers and SET Transmitters should consider the ramifications of sharing a particular subject identifier with a SET Recipient (e.g., whether doing so could enable correlation and/or de-anonymization of data) and choose appropriate subject identifiers for their use cases.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <https://www.rfc-editor.org/info/rfc7521>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8417] Hunt, P., Ed., Jones, M., Denniss, W., and M. Ansari, "Security Event Token (SET)", RFC 8417, DOI 10.17487/RFC8417, July 2018, <https://www.rfc-editor.org/info/rfc8417>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8935] Backman, A., Ed., Jones, M., Ed., Scurtescu, M., Ansari, M., and A. Nadalin, "Push-Based Security Event Token (SET) Delivery Using HTTP", RFC 8935, DOI 10.17487/RFC8935, November 2020, <https://www.rfc-editor.org/info/rfc8935>.

7.2. Informative References

[RFC6202] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", RFC 6202, DOI 10.17487/RFC6202, April 2011, <https://www.rfc-editor.org/info/rfc6202>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/info/rfc7617>.

Appendix A. Unencrypted Transport Considerations

Earlier versions of this specification made the use of TLS optional and described security and privacy considerations resulting from use of unencrypted HTTP as the underlying transport. When the working group decided to mandate usage of HTTP over TLS, it also decided to preserve the description of these considerations in a non-normative manner.

The considerations for using unencrypted HTTP with this protocol are the same as those described in Appendix A of [RFC8935], and are therefore not repeated here.

Acknowledgments

The editors would like to thank the members of the SCIM Working Group, which began discussions of provisioning events starting with draft-hunt-scim-notify-00 in 2015. We would like to thank Phil Hunt and the other authors of draft-ietf-secevent-delivery-02, upon which this specification is based. We would like to thank the participants in the SecEvents Working Group for their contributions to this specification.

Additionally, we would like to thank the following individuals for their reviews of this specification: Roman Danyliw, Martin Duke, Benjamin Kaduk, Erik Kline, Murray Kucherawy, Warren Kumari, Barry Leiba, Mark Nottingham, Alvaro Retana, Yaron Sheffer, Valery Smyslov, Robert Sparks, Éric Vyncke, and Robert Wilton.

Authors' Addresses

Annabelle Backman (EDITOR)
Amazon
Email: richanna@amazon.com

Michael B. Jones (EDITOR)
Microsoft
Email: mbj@microsoft.com
URI: https://self-issued.info/

Marius Scurtescu
Coinbase
Email: marius.scurtescu@coinbase.com

Morteza Ansari
Independent
Email: morteza@sharppics.com

Anthony Nadalin
Independent
Email: nadalin@prodigy.net