Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2020

A. Backman, Ed.
Amazon
M. Jones, Ed.
Microsoft
M. Scurtescu
Coinbase
M. Ansari
Cisco
A. Nadalin
Microsoft
June 24, 2020

Poll-Based Security Event Token (SET) Delivery Using HTTP
draft-ietf-secevent-http-poll-12

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 26, 2020.

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 [I-D.ietf-secevent-http-push].

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 [I-D.ietf-secevent-http-push].

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 2.1.

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 [I-D.ietf-secevent-http-push]. 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 returned first, or another selection algorithm MAY be used, such as prioritizing SETs in some manner that makes sense for the use case. first. 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 re-transmitted 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 which 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 which 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 which 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.
   eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFkOCIsImlhdCI6MTQ
   1ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiYXVkIjpbIm
   h0dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M
   2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIx
   ZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV
   2ZW50OmNyZWF0ZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcn
   MvNDRmNjE0MmRmOTZiZDZhYjYxZTc1MjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuY
   W1lIiwidXNlck5hbWUiLCJwYXNzd29yZCIsImVtYWlscyJdfX19.",
  "3d0c3cf797584bd193bd0fb1bd4e7d30":
   "eyJhbGciOiJub25lIn0.
   eyJqdGkiOiIzZDBjM2NmNzk3NTg0YmQxOTNiZDBmYjFiZDRlN2QzMCIsImlhdCI6MTQ
   1ODQ5NjAyNSwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiYXVkIjpbIm
   h0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M
   2I3NzU0IiwiaHR0cHM6Ly9qaHViLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIx
   ZDA4NjQxZDc2NzZlZTciXSwic3ViIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL1V
   zZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJldmVudHMiOnsidXJuOmlldG
   Y6cGFyYW1zOnNjaW06ZXZlbnQ6cGFzc3dvcmRSZXNldCI6eyJpZCI6IjQ0ZjYxNDJkZ
   jk2YmQ2YWI2MWU3NTIxZDkifSwiaHR0cHM6Ly9leGFtcGxlLmNvbS9zY2ltL2V2ZW50
   L3Bhc3N3b3JkUmVzZXRFeHQiOnsicmVzZXRBdHRlbXB0cyI6NX19fQ."
 }
}
```

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 Section 2.4.3 or Section 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
```

Example Poll Error Response

2.6. Error Response Handling

If a SET is invalid, error codes from the IANA "Security Event Token Delivery Error Codes" registry established by [I-D.ietf-secevent-http-push] are used in error responses. As

described in Section 2.3 of [I-D.ietf-secevent-http-push], 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 Delivery 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 Section 2.2 and Section 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 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 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 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 TLS and 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 specification requires no IANA actions.

7. References

7.1. Normative References

[I-D.ietf-secevent-http-push]
Backman, A., Jones, M., Scurtescu, M., Ansari, M., and A. Nadalin, "Push-Based Security Event Token (SET) Delivery Using HTTP", draft-ietf-secevent-http-push-12 (work in progress), June 2020.

[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>.

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 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 [I-D.ietf-secevent-http-push], and are therefore not repeated here.

Appendix B. Other Streaming Specifications

[[NOTE TO THE RFC EDITOR: This section to be removed prior to publication]]

A number of pub/sub, queuing, and streaming systems were reviewed as possible solutions or as input to the current draft. These are listed in Appendix B of [I-D.ietf-secevent-http-push], and are therefore not repeated here.

Appendix C. 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 the 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 the 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, Eric Vyncke, and Robert Wilton.

Appendix D. Change Log

[[to be removed by the RFC Editor before publication as an RFC]]

Draft 00 - AB - Based on draft-ietf-secevent-delivery-02 with the following additions:

- Renamed to "Poll-Based SET Token Delivery Using HTTP"
- Removed references to the HTTP Push delivery method.

Draft 01 - mbj:

- Addressed problems identified in my 18-Jul-18 review message titled "Issues for both the Push and Poll Specs".
- Changes to align terminology with RFC 8417, for instance, by using the already defined term SET Recipient rather than SET Receiver.
- Applied editorial and minor normative corrections.
- Updated Marius' contact information.
- Begun eliminating redundancies between this specification and "Push-Based Security Event Token (SET) Delivery Using HTTP" [I-D.ietf-secevent-http-push], referencing, rather that duplicating common normative text.

Draft 02 - mbj:

- Removed vestigial language remaining from when the push and poll delivery methods were defined in a common specification.
- Replaced remaining uses of the terms Event Transmitter and Event Recipient with the correct terms SET Transmitter and SET Recipient.
- Removed uses of the unnecessary term "Event Stream".
- Removed dependencies between the semantics of "maxEvents" and "returnImmediately".
- Said that PII in SETs is to be encrypted with TLS, JWE, or both.
- Corrected grammar and spelling errors.

Draft 03 - mbj:

- Corrected uses of "attribute" to "member" when describing JSON objects.
- Further alignment with the push draft.

Draft 04 - AB + mbj

- Referenced SET Transmitter definition in http-push.
- Removed incorrect normative text regarding SET construction.
- Consolidated general out-of-scope items under Introduction.
- Removed unnecessary HTTP headers in examples and added Content-Type.
- Added Content-Language requirement for error descriptions, aligning with http-push.
- Stated that bearer tokens SHOULD have a limited lifetime.
- Minor editorial fixes.

Draft 05 - AB + mbj

- Added normative text defining how to respond to invalid poll requests.
- Addressed shepherd comments by Yaron Sheffer.

Draft 06 - mbj

- Addressed nits identified by the idnits tool.

Draft 07 - mbj

- Addressed area director review comments by Benjamin Kaduk.

Draft 08 - mbj + AB

- Corrected editorial nits.

Draft 09 - AB

- Addressed area director review comments by Benjamin Kaduk:
  - Added text clarifying that determining the SET Recipient's service identity is out of scope.

* Removed unelaborated reference to use of authentication to prevent DoS attacks.

Draft 10 - mbj

- Addressed SecDir review comments by Valery Smyslov on draft-ietf-secevent-http-push-10 that also applied here.
- Addressed IETF last call comments by Mark Nottingham.
- Addressed GenArt review comments by Robert Sparks.

Draft 11 - mbj

- Revised to unambiguously require the use of TLS, while preserving descriptions of precautions needed for non-TLS use in an appendix.

Draft 12 - mbj

- Addressed IESG comments.

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
Cisco

Email: morteza.ansari@cisco.com

Anthony Nadalin
Microsoft

Email: tonynad@microsoft.com

Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 28, 2020

A. Backman, Ed.
Amazon
M. Jones, Ed.
Microsoft
M. Scurtescu
Coinbase
M. Ansari
Cisco
A. Nadalin
Microsoft

June 26, 2020

Push-Based Security Event Token (SET) Delivery Using HTTP
draft-ietf-secevent-http-push-14

Abstract

This specification defines how a Security Event Token (SET) can be delivered to an intended recipient using HTTP POST over TLS. The SET is transmitted in the body of an HTTP POST request to an endpoint operated by the recipient, and the recipient indicates successful or failed transmission via the HTTP response.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 28, 2020.

Table of Contents

1. Introduction and Overview

This specification defines a mechanism by which a transmitter of a Security Event Token (SET) [RFC8417] can deliver the SET to an intended SET Recipient via HTTP POST [RFC7231] over TLS. This is an alternative SET delivery method to the one defined in [I-D.ietf-secevent-http-poll].

Push-based SET delivery over HTTP POST is intended for scenarios where all of the following apply:

- The transmitter of the SET is capable of making outbound HTTP requests.
- The recipient is capable of hosting a TLS-enabled HTTP endpoint that is accessible to the transmitter.
- 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 the following terms defined in [RFC8417]: "Security Event Token (SET)", "SET Issuer", "SET Recipient", and "Event Payload", as well as the term defined below:

SET Transmitter An entity that delivers SETs in its possession to one or more SET Recipients.

2. SET Delivery

To deliver a SET to a given SET Recipient, the SET Transmitter makes a SET transmission request to the SET Recipient, with the SET itself contained within the request. The SET Recipient replies to this request with a response either acknowledging successful transmission

of the SET or indicating that an error occurred while receiving, parsing, and/or validating the SET.

Upon receipt of a SET, the SET Recipient SHALL validate that all of the following are true:

- The SET Recipient can parse the SET.
- The SET is authentic (i.e., it was issued by the issuer specified within the SET, and if signed, was signed by a key belonging to the issuer).
- The SET Recipient is identified as an intended audience of the SET.
- The SET Issuer is recognized as an issuer that the SET Recipient is willing to receive SETs from (e.g., the issuer is listed as allowed by the SET Recipient).
- The SET Recipient is willing to accept this SET from this SET Transmitter (e.g., the SET Transmitter is expected to send SETs with the issuer and subject of the SET in question).

The mechanisms by which the SET Recipient performs this validation are out of scope for this document. SET parsing, issuer identification, and audience identification are defined in [RFC8417]. The mechanism for validating the authenticity of a SET is deployment specific, and may vary depending on the authentication mechanisms in use, and whether the SET is signed and/or encrypted (See Section 3).

SET Transmitters MAY transmit SETs issued by another entity. The SET Recipient may accept or reject (i.e., return an error response such as "access_denied") a SET at its own discretion.

The SET Recipient persists the SET in a way that is sufficient to meet the SET Recipient’s own reliability requirements. The level and method of retention of SETs by SET Recipients is out of scope of this specification. Once the SET has been validated and persisted, the SET Recipient SHOULD immediately return a response indicating that the SET was successfully delivered. The SET Recipient SHOULD NOT perform further processing of the SET beyond the required validation steps prior to sending this response. Any additional steps SHOULD be executed asynchronously from delivery to minimize the time the SET Transmitter is waiting for a response.

The SET Transmitter MAY transmit the same SET to the SET Recipient multiple times, regardless of the response from the SET Recipient. The SET Recipient MUST respond as it would if the SET had not been

previously received by the SET Recipient. The SET Recipient MUST NOT expect or depend on a SET Transmitter to re-transmit a SET or otherwise make a SET available to the SET Recipient once the SET Recipient acknowledges that it was received successfully.

The SET Transmitter should not re-transmit a SET unless the SET Transmitter suspects that previous transmissions may have failed due to potentially recoverable errors (such as network outage or temporary service interruption at either the SET Transmitter or SET Recipient). In all other cases, the SET Transmitter SHOULD NOT re-transmit a SET. The SET Transmitter SHOULD delay retransmission for an appropriate amount of time to avoid overwhelming the SET Recipient (see Section 4).

2.1. Transmitting a SET

To transmit a SET to a SET Recipient, the SET Transmitter makes an HTTP POST request to a TLS-enabled HTTP endpoint provided by the SET Recipient. The "Content-Type" header field of this request MUST be "application/secevent+jwt" as defined in Sections 2.3 and 7.2 of [RFC8417], and the "Accept" header field MUST be "application/json". The request body MUST consist of the SET itself, represented as a JWT [RFC7519].

The SET Transmitter MAY include in the request an "Accept-Language" header field to indicate to the SET Recipient the preferred language(s) in which to receive error messages.

The mechanisms by which the SET Transmitter determines the HTTP endpoint to use when transmitting a SET to a given SET Recipient are not defined by this specification and are deployment specific.

The following is a non-normative example of a SET transmission request:

```
POST /Events HTTP/1.1
Host: notify.rp.example.com
Accept: application/json
Accept-Language: en-US, en;q=0.5
Content-Type: application/secevent+jwt

eyJ0eXAiOiJzZWNldmVudCtqd3QiLCJhbGciOiJIUzI1NiJ9Cg
.
eyJpc3MiOiJodHRwczovL2lkcC5leGFtcGxlLmNvbS8iLCJqdGkiOiI3NTZFNjk
3MTc1NjUyMDY5NjQ2NTZFNzQ2OTY2Njk2NTcyIiwiaWF0IjoxNTA4MTg0ODQ1LC
JhdWQiOiI2MzZDNjk2NTZFNzQ1RjY5NjQiLCJldmVudHMiOnsiaHR0cHM6Ly9zY
2hlbWFzLm9wZW5pZC5uZXQvc2VjZXZlbnQvcmlzYy9ldmVudC10eXBlL2FjY291
bnQtZGlzYWJsZWQiOnsic3ViamVjdCI6eyJzdWJqZWN0X3R5cGUiOiJpc3Mtc3V
iIiwiaXNzIjoiaHR0cHM6Ly9pZHAuZXhhbXBsZS5jb20vIiwic3ViIjoiNzM3NT
YyNkE2NTYzNzQifSwicmVhc29uIjoiaGlqYWNraW5nIn19fQ
.
Y4rXxMD406P2edv00cr9Wf3_XwNtLjB9n-jTqN1_lLc
```

Figure 1: Example SET Transmission Request

2.2. Success Response

If the SET is determined to be valid, the SET Recipient SHALL acknowledge successful transmission by responding with HTTP Response Status Code 202 (Accepted) (see Section 6.3.3 of [RFC7231]). The body of the response MUST be empty.

The following is a non-normative example of a successful receipt of a SET.

```
HTTP/1.1 202 Accepted
```

Figure 2: Example Successful Delivery Response

2.3. Failure Response

In the event of a general HTTP error condition, the SET Recipient responds with the applicable HTTP Status Code, as defined in Section 6 of [RFC7231].

When the SET Recipient detects an error parsing, validating, or authenticating a SET transmitted in a SET Transmission Request, the SET Recipient SHALL respond with an HTTP Response Status Code of 400 (Bad Request). The "Content-Type" header field of this response MUST

be "application/json", and the body MUST be a UTF-8 encoded JSON [RFC8259] object containing the following name/value pairs:

err A Security Event Token Error Code (see Section 2.4).

description A UTF-8 string containing a human-readable description of the error that may provide additional diagnostic information. The exact content of this field is implementation specific.

The response MUST include a "Content-Language" header field, whose value indicates the language of the error descriptions included in the response body. If the SET Recipient can provide error descriptions in multiple languages, they SHOULD choose the language to use according to the value of the "Accept-Language" header field sent by the SET Transmitter in the transmission request, as described in Section 5.3.5 of [RFC7231]. If the SET Transmitter did not send an "Accept-Language" header field, or if the SET Recipient does not support any of the languages included in the header field, the SET Recipient MUST respond with messages that are understandable by an English-speaking person, as described in Section 4.5 of [RFC2277].

The following is a non-normative example error response indicating that the key used to encrypt the SET has been revoked.

```
HTTP/1.1 400 Bad Request
Content-Language: en-US
Content-Type: application/json

{
  "err": "invalid_key",
  "description": "Key ID 12345 has been revoked."
}
```

Figure 3: Example Error Response (invalid_key)

The following is a non-normative example error response indicating that the access token included in the request is expired.

```
HTTP/1.1 400 Bad Request
Content-Language: en-US
Content-Type: application/json

{
  "err": "authentication_failed",
  "description": "Access token has expired."
}
```

Figure 4: Example Error Response (authentication_failed)

The following is a non-normative example error response indicating that the SET Receiver is not willing to accept SETs issued by the specified issuer from this particular SET Transmitter.

```
HTTP/1.1 400 Bad Request
Content-Language: en-US
Content-Type: application/json

{
  "err": "invalid_issuer",
  "description": "Not authorized for issuer https://iss.example.com/."
}
```

Figure 5: Example Error Response (access_denied)

2.4. Security Event Token Delivery Error Codes

Security Event Token Delivery Error Codes are strings that identify a specific category of error that may occur when parsing or validating a SET. Every Security Event Token Delivery Error Code MUST have a unique name registered in the IANA "Security Event Token Delivery Error Codes" registry established by Section 7.1.

The following table presents the initial set of Error Codes that are registered in the IANA "Security Event Token Delivery Error Codes" registry:

Error Code	Description
invalid_request	The request body cannot be parsed as a SET, or the Event Payload within the SET does not conform to the event's definition.
invalid_key	One or more keys used to encrypt or sign the SET is invalid or otherwise unacceptable to the SET Recipient (expired, revoked, failed certificate validation, etc.).
invalid_issuer	The SET issuer is invalid for the SET Recipient.
invalid_audience	The SET audience does not correspond to the SET Recipient.
authentication_failed	The SET Recipient could not authenticate the SET Transmitter.
access_denied	The SET Transmitter is not authorized to transmit the SET to the SET Recipient.

Table 1: SET Delivery Error Codes

Other Error Codes may also be received, as the set of Error Codes is extensible via the IANA "Security Event Token Delivery Error Codes" registry established in Section 7.1.

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 DANE [RFC6698].

Authorization for the eligibility to provide actionable SETs can be determined by using the identity of the SET Issuer, the identity of the SET Transmitter, perhaps using mutual TLS, or via other employed authentication methods. Because SETs are not commands, SET Recipients are free to ignore SETs that are not of interest.

4. Delivery Reliability

Delivery reliability requirements may vary depending upon the use cases. This specification defines the response from the SET Recipient in such a way as to provide the SET Transmitter with the information necessary to determine what further action is required, if any, in order to meet their requirements. SET Transmitters with

high reliability requirements may be tempted to always retry failed transmissions. However, it should be noted that for many types of SET delivery errors, a retry is extremely unlikely to be successful. For example, "invalid_request" indicates a structural error in the content of the request body that is likely to remain when re-transmitting the same SET. Others such as "access_denied" may be transient, for example if the SET Transmitter refreshes expired credentials prior to re-transmission.

The SET Transmitter may be unaware of whether or not a SET has been delivered to a SET Recipient. For example, a network interruption could prevent the SET Transmitter from receiving the success response, or a service outage could prevent the SET Transmitter from recording the fact that the SET was delivered. It is left to the implementer to decide how to handle such cases, based on their requirements. For example, it may be appropriate for the SET Transmitter to re-transmit the SET to the SET Recipient, erring on the side of guaranteeing delivery, or it may be appropriate to assume delivery was successful, erring on the side of not spending resources re-transmitting previously delivered SETs. Other options, such as sending the SET to a "dead letter queue" for manual examination may also be appropriate.

Implementers SHOULD evaluate the reliability requirements of their use cases and the impact of various retry mechanisms and re-transmission policies on the performance of their systems to determine an appropriate strategy for handling various error conditions.

5. Security Considerations

5.1. 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.

5.2. HTTP Considerations

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

5.3. 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. TLS MUST be used to secure the transmitted SETs. In some use cases, encrypting the SET as described in 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 Transmitter determines the expected service identity to match the SET Recipient's server certificate against is out of scope for this document. The implementation security considerations for TLS in "Recommendations for Secure Use of TLS and DTLS" [RFC7525] MUST be followed.

5.4. Denial of Service

The SET Recipient may be vulnerable to a denial-of-service attack where a malicious party makes a high volume of requests containing invalid SETs, causing the endpoint to expend significant resources on cryptographic operations that are bound to fail. This may be mitigated by authenticating SET Transmitters with a mechanism such as mutual TLS. Rate-limiting problematic transmitters is also a possible means of mitigation.

5.5. Authenticating Persisted SETs

At the time of receipt, the SET Recipient can rely upon TLS mechanisms, HTTP authentication methods, and/or other context from the transmission request to authenticate the SET Transmitter and validate the authenticity of the SET. However, this context is typically unavailable to systems to which the SET Recipient forwards the SET, or to systems that retrieve the SET from storage. If the SET Recipient requires the ability to validate SET authenticity outside of the context of the transmission request, then the SET Recipient SHOULD ensure that such SETs have been signed in accordance with [RFC7515]. Needed context could also be stored with the SET and retrieved with it.

6. 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.

7. IANA Considerations

7.1. Security Event Token Delivery Error Codes

This document defines Security Event Token Delivery Error Codes, for which IANA is asked to create and maintain a new registry titled "Security Event Token Delivery Error Codes". Initial values for the Security Event Token Delivery Error Codes registry are defined in Table 1 and registered below. Future assignments are to be made through the Specification Required registration policy ([RFC8126]) and shall follow the template below.

Error Codes are intended to be interpreted by automated systems, and therefore SHOULD identify classes of errors to which an automated system could respond in a meaningfully distinct way (e.g., by refreshing authentication credentials and retrying the request).

Error Code names are case sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration description is clear.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

7.1.1. Registration Template

Error Code
: The name of the Security Event Token Delivery Error Code, as described in Section 2.4. The name MUST be a case-sensitive ASCII string consisting only of letters, digits, and underscore; these are the characters whose codes fall within the inclusive ranges 0x30-39, 0x41-5A, 0x5F and 0x61-7A.

Description
: A brief human-readable description of the Security Event Token Delivery Error Code.

Change Controller
: For error codes registered by the IETF or its working groups, list "IETF". For all other error codes, list the name of the party responsible for the registration. Contact information such as mailing address, email address, or phone number may also be provided.

Defining Document(s)
: A reference to the document or documents that define the Security Event Token Delivery Error Code. The definition MUST specify the name and description of the error code and explain under what circumstances the error code may be used. URIs that can be used to retrieve copies of each document at no cost SHOULD be included.

7.1.2. Initial Registry Contents

Error Code: invalid_request
Description: The request body cannot be parsed as a SET or the event payload within the SET does not conform to the event's definition.
Change Controller: IETF
Defining Document(s): Section 2.4 of [[this specification]]

Error Code: invalid_key
Description: One or more keys used to encrypt or sign the SET is invalid or otherwise unacceptable to the SET Recipient (expired, revoked, failed certificate validation, etc.).
Change Controller: IETF
Defining Document(s): Section 2.4 of [[this specification]]

Error Code: invalid_issuer
Description: The SET issuer is invalid for the SET Recipient.
Change Controller: IETF
Defining Document(s): Section 2.4 of [[this specification]]

Error Code: invalid_audience
Description: The SET audience does not correspond to the SET Recipient.
Change Controller: IETF
Defining Document(s): Section 2.4 of [[this specification]]

Error Code: authentication_failed
Description: The SET Recipient could not authenticate the SET Transmitter.
Change Controller: IETF
Defining Document(s): Section 2.4 of [[this specification]]

Error Code: access_denied
Description: The SET Transmitter is not authorized to transmit the SET to the SET Recipient.
Change Controller: IETF
Defining Document(s): Section 2.4 of [[this specification]]

8. References

8.1. 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>.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, DOI 10.17487/RFC2277, January 1998, <https://www.rfc-editor.org/info/rfc2277>.

[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>.

[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>.

[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>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

8.2. Informative References

[I-D.ietf-secevent-http-poll] Backman, A., Jones, M., Scurtescu, M., Ansari, M., and A. Nadalin, "Poll-Based Security Event Token (SET) Delivery Using HTTP", draft-ietf-secevent-http-poll-12 (work in progress), June 2020.

[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>.

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 HTTP over TLS, it also decided to preserve the description of these considerations in this non-normative appendix.

SETs may contain sensitive information that is considered Personally Identifiable Information (PII). In such cases, SET Transmitters and SET Recipients MUST protect the confidentiality of the SET contents. When TLS is not used, this means that the SET MUST be encrypted as described in JWE [RFC7516].

If SETs were allowed to be transmitted over unencrypted channels, some privacy-sensitive information about them might leak, even though the SETs themselves are encrypted. For instance, an attacker may be able to determine whether or not a SET was accepted and the reason for its rejection or may be able to derive information from being able to observe the size of the encrypted SET. (Note that even when TLS is utilized, some information leakage is still possible; message padding algorithms to prevent side channels remain an open research topic.)

Appendix B. Other Streaming Specifications

[[NOTE TO THE RFC EDITOR: This section to be removed prior to publication]]

The following pub/sub, queuing, and streaming systems were reviewed as possible solutions or as input to the current draft:

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

In addition to this specification, the WG is defining a polling-based SET delivery protocol. That protocol [I-D.ietf-secevent-http-poll] describes it as:

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.

XMPP Events

The WG considered XMPP Events and their ability to provide a single messaging solution without the need for both polling and push modes. The feeling was the size and methodology of XMPP was too far apart from the current capabilities of the SECEVENTs community, which focuses in on HTTP based service delivery and authorization.

Amazon Simple Notification Service

Simple Notification Service is a pub/sub messaging product from AWS. SNS supports a variety of subscriber types: HTTP/HTTPS endpoints, AWS Lambda functions, email addresses (as JSON or plain text), phone numbers (via SMS), and AWS SQS standard queues. It does not directly support pull, but subscribers can get the pull model by creating an SQS queue and subscribing it to the topic. Note that this puts the cost of pull support back onto the subscriber, just as it is in the push model. It is not clear that one way is strictly better than the other; larger, sophisticated developers may be happy to own message persistence so they can have their own internal delivery guarantees. The long tail of OIDC clients may not care about that or may fail to get it right. Regardless, I think we can learn something from the Delivery Policies supported by SNS, as well as the delivery controls that SQS offers (e.g., Visibility Timeout, Dead-Letter Queues). I am not suggesting that we need all of these things in the spec, but they give an idea of what features people have found useful.

Other information:

- API Reference:
  http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/Welcome.html

- Visibility Timeouts:
  http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html

Apache Kafka

Apache Kafka is an Apache open source project based upon TCP for distributed streaming. It prescribes some interesting general-purpose features that seem to extend far beyond the simpler streaming model that SECEVENTs is after. A comment from MS has been that Kafka does an acknowledge with poll combination event which seems to be a performance advantage. See: https://kafka.apache.org/intro

Google Pub/Sub

The Google Pub Sub system favors a model whereby polling and acknowledgement of events is done with separate endpoints and as separate functions.

Information:

- Cloud Overview - https://cloud.google.com/pubsub/

- Subscriber Overview - https://cloud.google.com/pubsub/docs/subscriber

- Subscriber Pull(poll) - https://cloud.google.com/pubsub/docs/pull

Appendix C. 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 the specification: Joe Clarke, Roman Danyliw, Vijay Gurbani, Benjamin Kaduk, Erik Kline, Murray Kucherawy, Barry Leiba, Yaron Sheffer, Robert Sparks, Valery Smyslov, Eric Vyncke, and Robert Wilton.

Appendix D. Change Log

[[NOTE TO THE RFC EDITOR: This section to be removed prior to publication]]

Draft 00 - AB - Based on draft-ietf-secevent-delivery-02 with the following changes:

- Renamed to "Push-Based SET Token Delivery Using HTTP"
- Removed references to the HTTP Polling delivery method.
- Removed informative reference to RFC6202.

Draft 01 - AB:

- Fixed area and workgroup to match secevent.
- Removed unused definitions and definitions already covered by SET.
- Renamed Event Transmitter and Event Receiver to SET Transmitter and SET Receiver, respectively.
- Added IANA registry for SET Delivery Error Codes.
- Removed enumeration of HTTP authentication methods.
- Removed generally applicable guidance for HTTP, authorization tokens, and bearer tokens.
- Moved guidance for using authentication methods as DoS protection to Security Considerations.
- Removed redundant instruction to use WWW-Authenticate header.
- Removed further generally applicable guidance for authorization tokens.
- Removed bearer token from example delivery request, and text referencing it.
- Broke delivery method description into separate request/response sections.
- Added missing empty line between headers and body in example request.
- Removed inapplicable notes about example formatting.

- Removed text about SET creation and handling.
- Removed duplication in protocol description.
- Added "non-normative example" text to example transmission request.
- Fixed inconsistencies in use of Error Code term.

Draft 02 - AB:

- Rewrote abstract and introduction.
- Rewrote definitions for SET Transmitter, SET Receiver.
- Renamed Event Delivery section to SET Delivery.
- Readability edits to Success Response and Failure Response sections.
- Consolidated definition of error response under Failure Response section.
- Removed Event Delivery Process section and moved its content to parent section.
- Readability edits to SET Delivery section and its subsections.
- Added callout that SET Receiver HTTP endpoint configuration is out-of-scope.
- Added callout that SET verification mechanisms are out-of-scope.
- Added retry guidance, notes regarding delivery reliability requirements.
- Added guidance around using JWS and/or JWE to authenticate persisted SETs.

Draft 03 - mbj:

- Addressed problems identified in my 18-Jul-18 review message titled "Issues for both the Push and Poll Specs".
- Changes to align terminology with RFC 8417, for instance, by using the already defined term SET Recipient rather than SET Receiver.
- Applied editorial and minor normative corrections.

- Updated Marius' contact information.

Draft 04 - AB:

- Replaced Error Codes with smaller set of meaningfully differentiated codes.
- Added more error response examples.
- Removed un-referenced normative references.
- Added normative reference to JSON in error response definition.
- Added text clarifying that the value of the "description" attribute in error responses is implementation specific.
- Added requirement that error descriptions and responses are UTF-8 encoded.
- Added error description language preferences and specification via "Accept-Language" and "Content-Language" headers.
- Added "recognized issuer" validation requirement in section 2.
- Added timeouts as an acceptable reason to resend a SET in section 2.
- Edited text in section 1 to clarify that configuration is out of scope.
- Made minor editorial corrections.

Draft 05 - AB:

- Made minor editorial corrections.
- Updated example request with a correct SET header and signature.
- Revised TLS guidance to allow implementers to provide confidentiality protection via JWE.
- Revised TLS guidance to require *at least* TLS 1.2.
- Revised TLS guidance to recommend supporting the newest version of TLS that meets security requirements.
- Revised SET Delivery Error Code format to allow the same set of characters as is allowed in error codes in RFC6749.

- Added mention of HTTP Poll spec to list of other streaming specs in appendix.
- Added validation step requiring SET Recipient to verify that the SET is one which the SET Transmitter is expected to send to the SET Recipient.
- Changed responding to errors with an appropriate HTTP status code from optional to recommended.
- Changed Error Codes registry change policy from Expert Review to First Come First Served; added guidance that error codes are meant to be consumed by automated systems.
- Added text making clear that it is up to SET Recipients whether or not they will accept SETs where the SET Issuer is different from the SET Transmitter.
- Reworded guidance around signing and/or encrypting SETs for integrity protection.
- Renamed TLS "Support Considerations" section to "Confidentiality of SETs".
- Reworded guidance around subject identifier selection and privacy concerns.

Draft 06 - mbj, MS:

- Made minor editorial corrections.
- Updated to indicate that failure response should be returned if errors occur in authenticating the SET.
- Updated reference for JSON from RFC 7159 to RFC 8259.
- Fixed Authentication Using Signed SETs to indicate the SET Transmitter must be authorized to deliver the SET, not the SET Issuer.
- Fixed Authenticating Persisted SETs to put the responsibility for ensuring the SET is signed on the SET Recipient.
- Fixed error code format definition to match error codes defined in doc.

Draft 07 - AB:

- Made minor editorial corrections.
- Removed "SET Recipient" definition and added explicit list of terms used from RFC8417.

Draft 08 - mbj

- Addressed area director review comments by Benjamin Kaduk.

Draft 09 - mbj + AB

- Corrected editorial nits.

Draft 10 - AB

- Addressed area director review comments by Benjamin Kaduk:
  - Added reference to 8417 as definition document for SETs.
  - Added text clarifying that determining the SET Recipient's service identity is out of scope.
  - Added normative recommendation for transmitters to target SETs to specific business needs of subscribers.
  - Minor editorial corrections.

Draft 11 - mbj

- Addressed SecDir review comments by Valery Smyslov.
- Addressed OpsDir review comments by Joe Clarke.
- Addressed GenArt review comments by Vijay Gurbani.

Draft 12 - mbj

- Revised to unambiguously require the use of TLS, while preserving descriptions of precautions needed for non-TLS use in an appendix.

Draft 13 - mbj

- Addressed IESG comments.

Draft 14 - AB

- Revised normative requirements for SET re-transmission to clarify "at least once" delivery expectiations.

- Added non-normative text to Section 4 - Delivery Reliability describing conditions where re-transmission of successfully delivered SETs may occur.

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
Cisco

Email: morteza.ansari@cisco.com

Anthony Nadalin
Microsoft

Email: tonynad@microsoft.com

Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 8, 2021

A. Backman, Ed.
Amazon
M. Scurtescu
Coinbase

September 04, 2020

Subject Identifiers for Security Event Tokens
draft-ietf-secevent-subject-identifiers-06

Abstract

Security events communicated within Security Event Tokens may support a variety of identifiers to identify the subject and/or other principals related to the event. This specification formalizes the notion of subject identifiers as named sets of well-defined claims describing the subject, a mechanism for representing subject identifiers within a JSON object such as a JSON Web Token (JWT) or Security Event Token (SET), and a registry for defining and allocating names for these claim sets.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 8, 2021.

Table of Contents

1. Introduction

As described in Section 1.2 of SET [RFC8417], the subject of a security event may take a variety of forms, including but not limited to a JWT [RFC7519] principal, an IP address, a URL, etc. Furthermore, even in the case where the subject of an event is more narrowly scoped, there may be multiple ways by which a given subject may be identified. For example, an account may be identified by an

opaque identifier, an email address, a phone number, a JWT "iss" claim and "sub" claim, etc., depending on the nature and needs of the transmitter and receiver. Even within the context of a given transmitter and receiver relationship, it may be appropriate to identify different accounts in different ways, for example if some accounts only have email addresses associated with them while others only have phone numbers. Therefore it can be necessary to indicate within a SET the mechanism by which the subject of the security event is being identified.

To address this problem, this specification defines Subject Identifiers - JSON [RFC7159] objects containing information identifying a subject - and Subject Identifier Types - named sets of rules describing how to encode different kinds of subject identifying information (e.g., an email address, or an issuer and subject pair) as a Subject Identifier.

Below is a non-normative example of a Subject Identifier that identifies a subject by email address, using the Email Subject Identifier Type.

```
{
  "subject_type": "email",
  "email": "user@example.com",
}
```

Figure 1: Example: Subject Identifier using the Email Subject Identifier Type

Subject Identifiers are intended to be a general purpose mechanism for identifying principals within JSON objects. Below is a non-normative example of a JWT that uses a Subject Identifier in the "sub_id" claim (defined in this specification) to identify its subject.

```
{
  "iss": "issuer.example.com",
  "sub_id": {
    "subject_type": "phone_number",
    "phone_number": "+12065550100",
  },
}
```

Figure 2: Example: JWT using a Subject Identifier with the sub_id claim

Below is a non-normative example of a SET containing a hypothetical security event describing the interception of a message, using

Subject Identifiers to identify the sender, intended recipient, and interceptor.

```
{
  "iss": "issuer.example.com",
  "iat": 1508184845,
  "aud": "aud.example.com",
  "events": {
    "https://secevent.example.com/events/message-interception": {
      "from": {
        "subject_type": "email",
        "email": "alice@example.com",
      },
      "to": {
        "subject_type": "email",
        "email": "bob@example.com",
      },
      "interceptor": {
        "subject_type": "email",
        "email": "eve@example.com",
      },
    },
  },
}
```

Figure 3: Example: SET with an event payload containing multiple Subject Identifiers

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.1. Definitions

This specification utilizes terminology defined in [RFC7159], [RFC7519], and [RFC8417].

3. Subject Identifiers

A Subject Identifier is a JSON [RFC7159] object whose contents may be used to identify a principal within some context. A Subject Identifier Type is a named definition of a set of information that may be used to identify a principal, and the rules for encoding that information as a Subject Identifier. A Subject Identifier MUST conform to a specific Subject Identifier Type, and MUST contain a

"subject_type" member whose value is the name of that Subject Identifier Type.

Every Subject Identifier Type MUST have a unique name registered in the IANA "Security Event Subject Identifier Types" registry established by Section 7.1, or a Collision-Resistant Name as defined in [RFC7519]. Subject Identifier Types that are expected to be used broadly by a variety of parties SHOULD be registered in the "Security Event Subject Identifier Types" registry.

A Subject Identifier Type MAY describe more members than are strictly necessary to identify a subject, and MAY describe conditions under which those members are required, optional, or prohibited.

Aside from the "subject_type" member whose definition is given above, every member within a Subject Identifier MUST match the format specified for that member by the Subject Identifier's Subject Identifier Type. A Subject Identifier MUST NOT contain any members prohibited or not described by its Subject Identifier Type, and MUST contain all members required by its Subject Identifier Type.

3.1. Subject Identifier Types versus Principal Types

A Subject Identifier Type describes a way to identify a principal, but does not explicitly indicate the type of that principal (e.g., user, group, network connection, baseball team, astronomic object). Consequently Subject Identifiers remove ambiguity around how a principal is being identified, and how to parse an identifying structure, but they do not remove ambiguity around how to resolve that identifier to a principal. For example, consider a directory management API that allows callers to identify users and groups through both immutable unique identifiers and mutable email addresses. Such an API could use Subject Identifiers to disambiguate between which of these two types of identifiers is in use. However, the service would have to determine whether the principal is a user or group via some other means, such as by querying a database or by inferring the type from the API contract.

3.2. Subject Identifier Type Definitions

The following Subject Identifier Types are registered in the IANA "Security Event Subject Identifier Types" registry established by Section 7.1.

3.2.1. Account Subject Identifier Type

The Account Subject Identifier Type identifies a principal using an account at a service provider, identified with an "acct" URI as defined in [RFC7565]. Subject Identifiers of this type MUST contain a "uri" member whose value is the "acct" URI for the subject. The "uri" member is REQUIRED and MUST NOT be null or empty. The Account Subject Identifier Type is identified by the name "account".

Below is a non-normative example Subject Identifier for the Account Subject Identifier Type:

```
{
  "subject_type": "account",
  "uri": "acct:example.user@service.example.com",
}
```

Figure 4: Example: Subject Identifier for the Account Subject Identifier Type

3.2.2. Email Subject Identifier Type

The Email Subject Identifier Type identifies a principal using an email address. Subject Identifiers of this type MUST contain an "email" member whose value is a string containing the email address of the principal, formatted as an "addr-spec" as defined in Section 3.4.1 of [RFC5322]. The "email" member is REQUIRED and MUST NOT be null or empty. The value of the "email" member SHOULD identify a mailbox to which email may be delivered, in accordance with [RFC5321]. The Email Subject Identifier Type is identified by the name "email".

Below is a non-normative example Subject Identifier for the Email Subject Identifier Type:

```
{
  "subject_type": "email",
  "email": "user@example.com",
}
```

Figure 5: Example: Subject Identifier for the Email Subject Identifier Type

3.2.2.1. Email Canonicalization

Many email providers will treat multiple email addresses as equivalent. While the domain portion of an [RFC5322] email address is consistently treated as case-insensitive per [RFC1034], some

providers treat the local part of the email address as case-insensitive as well, and consider "user@example.com", "User@example.com", and "USER@example.com" as the same email address. This has led users to view these strings as equivalent, driving service providers to implement proprietary email canonicalization algorithms to ensure that email addresses entered by users resolve to the same canonical string. When receiving an Email Subject Identifier, the recipient SHOULD use their implementation's canonicalization algorithm to resolve the email address to the same string used in their system.

3.2.3. Phone Number Subject Identifier Type

The Phone Number Subject Identifier Type identifies a principal using a telephone number. Subject Identifiers of this type MUST contain a "phone_number" member whose value is a string containing the full telephone number of the principal, including international dialing prefix, formatted according to E.164 [E164]. The "phone_number" member is REQUIRED and MUST NOT be null or empty. The Phone Number Subject Identifier Type is identified by the name "phone_number".

Below is a non-normative example Subject Identifier for the Email Subject Identifier Type:

```
{
  "subject_type": "phone_number",
  "phone_number": "+12065550100",
}
```

Figure 6: Example: Subject Identifier for the Phone Number Subject Identifier Type.

3.2.4. Issuer and Subject Subject Identifier Type

The Issuer and Subject Subject Identifier Type identifies a principal using a pair of "iss" and "sub" members, analagous to how subjects are identified using the "iss" and "sub" claims in OpenID Connect [OpenID.Core] ID Tokens. These members MUST follow the formats of the "iss" member and "sub" member defined by [RFC7519], respectively. Both the "iss" member and the "sub" member are REQUIRED and MUST NOT be null or empty. The Issuer and Subject Subject Identifier Type is identified by the name "iss_sub".

Below is a non-normative example Subject Identifier for the Issuer and Subject Subject Identifier Type:

```
{
  "subject_type": "iss_sub",
  "iss": "http://issuer.example.com/",
  "sub": "145234573",
}
```

Figure 7: Example: Subject Identifier for the Issuer and Subject Subject Identifier Type

3.2.5. Aliases Subject Identifier Type

The Aliases Subject Identifier Type describes a subject that is identified with a list of different Subject Identifiers. It is intended for use when a variety of identifiers have been shared with the party that will be interpreting the Subject Identifier, and it is unknown which of those identifiers they will recognize or support. Subject Identifiers of this type MUST contain an "identifiers" member whose value is a JSON array containing one or more Subject Identifiers. Each Subject Identifier in the array MUST identify the same entity. The "identifiers" member is REQUIRED and MUST NOT be null or empty. It MAY contain multiple instances of the same Subject Identifier Type (e.g., multiple Email Subject Identifiers), but SHOULD NOT contain exact duplicates. This type is identified by the name "aliases".

"alias" Subject Identifiers MUST NOT be nested; i.e., the "identifiers" member of an "alias" Subject Identifier MUST NOT contain a Subject Identifier of type "aliases".

Below is a non-normative example Subject Identifier for the Aliases Subject Identifier Type:

```
{
  "subject_type": "aliases",
  "identifiers": [
    {
      "subject_type": "email",
      "email": "user@example.com",
    },
    {
      "subject_type": "phone_number",
      "phone_number": "+12065550100",
    },
    {
      "subject_type": "email",
      "email": "user+qualifier@example.com",
    }
  ],
}
```

Figure 8: Example: Subject Identifier for the Aliases Subject Identifier Type

4. Subject Identifiers in JWTs

4.1. "sub_id" Claim

The "sub" JWT Claim is defined in Section 4.1.2 of [RFC7519] as containing a string value, and therefore cannot contain a Subject Identifier (which is a JSON object) as its value. This document defines the "sub_id" JWT Claim, in accordance with Section 4.2 of [RFC7519], as a common claim that identifies the subject of the JWT using a Subject Identifier. When present, the value of this claim MUST be a Subject Identifier that identifies the principal that is the subject of the JWT. The "sub_id" claim MAY be included in a JWT, whether or not the "sub" claim is present. When both the "sub" and "sub_id" claims are present in a JWT, they MUST identify the same principal.

When processing a JWT with both "sub" and "sub_id" claims, implementations MUST NOT rely on both claims to determine the subject. An implementation MAY attempt to determine the subject from one claim and fall back to using the other if it determines it does not understand the format of the first claim. For example, an implementation may attempt to use "sub_id", and fall back to using "sub" upon finding that "sub_id" contains a Subject Identifier whose type is not recognized by the implementation.

Below are non-normative examples of JWTs containing the "sub_id" claim:

```
{
  "iss": "issuer.example.com",
  "sub_id": {
    "subject_type": "email",
    "email": "user@example.com",
  },
}
```

Figure 9: Example: JWT containing a 'sub_id' claim and no 'sub' claim

```
{
  "iss": "issuer.example.com",
  "sub": "user@example.com",
  "sub_id": {
    "subject_type": "email",
    "email": "user@example.com",
  },
}
```

Figure 10: Example: JWT where both the 'sub' and 'sub_id' claims identify the subject using the same identifier

```
{
  "iss": "issuer.example.com",
  "sub": "user@example.com",
  "sub_id": {
    "subject_type": "email",
    "email": "elizabeth@example.com",
  },
}
```

Figure 11: Example: JWT where both the 'sub' and 'sub_id' claims identify the subject using different values of the same identifier type

```
{
  "iss": "issuer.example.com",
  "sub": "user@example.com",
  "sub_id": {
    "subject_type": "account",
    "uri": "acct:example.user@service.example.com",
  },
}
```

Figure 12: Example: JWT where the 'sub' and 'sub_id' claims identify the subject via different types of identifiers

4.2. "sub_id" and "iss_sub" Subject Identifiers

The "sub_id" claim MAY contain an "iss_sub" Subject Identifier. In this case, the JWT's "iss" claim and the Subject Identifier's "iss" member MAY be different. For example, an OpenID Connect [OpenID.Core] client may construct such a JWT when issuing a JWT back to its OpenID Connect Identity Provider, in order to communicate information about the services' shared subject principal using an identifier the Identity Provider is known to understand. Similarly, the JWT's "sub" claim and the Subject Identifier's "sub" member MAY be different. For example, this may be used by an OpenID Connect client to communicate the subject principal's local identifier at the client back to its Identity Provider.

Below are non-normative examples of a JWT where the "iss" claim and "iss" member within the "sub_id" claim are the same, and a JWT where they are different.

```
{
  "iss": "issuer.example.com",
  "sub_id": {
    "subject_type": "iss_sub",
    "iss": "issuer.example.com",
    "sub": "example_user",
  },
}
```

Figure 13: Example: JWT with a `iss_sub` Subject Identifier where JWT issuer and subject issuer are the same

```
{
  "iss": "client.example.com",
  "sub_id": {
    "subject_type": "iss_sub",
    "iss": "issuer.example.com",
    "sub": "example_user",
  },
}
```

Figure 14: Example: JWT with an `iss_sub` Subject Identifier where the JWT issuer and subject issuer are different

```
{
  "iss": "client.example.com",
  "sub": "client_user",
  "sub_id": {
    "subject_type": "iss_sub",
    "iss": "issuer.example.com",
    "sub": "example_user",
  },
}
```

Figure 15: Example: JWT with an ‘iss_sub‘ Subject Identifier where the JWT ‘iss‘ and ‘sub‘ claims differ from the Subject Identifier’s ‘iss‘ and ‘sub‘ members

5. Privacy Considerations

5.1. Identifier Correlation

The act of presenting two or more identifiers for a single principal together (e.g., within an "aliases" Subject Identifier, or via the "sub" and "sub_id" JWT claims) may communicate more information about the principal than was intended. For example, the entity to which the identifiers are presented, now knows that both identifiers relate to the same principal, and may be able to correlate additional data based on that. When transmitting Subject Identifiers, the transmitter SHOULD take care that they are only transmitting multiple identifiers together when it is known that the recipient already knows that the identifiers are related (e.g., because they were previously sent to the recipient as claims in an OpenID Connect ID Token), or when correlation is essential to the use case.

The considerations described in Section 6 of [RFC8417] also apply when Subject Identifiers are used within SETs. The considerations described in Section 12 of [RFC7519] also apply when Subject Identifiers are used within JWTs.

6. Security Considerations

6.1. Confidentiality and Integrity

This specification does not define any mechanism for ensuring the confidentiality or integrityi of a Subject Identifier. Where such properties are required, implementations MUST use mechanisms provided by the containing format (e.g., integrity protecting SETs or JWTs using JWS [RFC7515]), or at the transport layer or other layer in the application stack (e.g., using TLS [RFC8446]).

Further considerations regarding confidentiality and integrity of SETs can be found in Section 5.1 of [RFC8417].

7. IANA Considerations

7.1. Security Event Subject Identifier Types Registry

This document defines Subject Identifier Types, for which IANA is asked to create and maintain a new registry titled "Security Event Subject Identifier Types". Initial values for the Security Event Subject Identifier Types registry are given in Section 3. Future assignments are to be made through the Expert Review registration policy [BCP26] and shall follow the template presented in Section 7.1.2.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

7.1.1. Registry Location

(This section to be removed by the RFC Editor before publication as an RFC.)

The authors recommend that the Subject Identifier Types registry be located at "https://www.iana.org/assignments/secevent/".

7.1.2. Registration Template

Type Name
: The name of the Subject Identifier Type, as described in Section 3. The name MUST be an ASCII string consisting only of lower-case characters ("a" - "z"), digits ("0" - "9"), underscores ("_"), and hyphens ("-"), and SHOULD NOT exceed 20 characters in length.

Type Description
: A brief description of the Subject Identifier Type.

Change Controller
: For types defined in documents published by the IETF or its working groups, list "IETF". For all other types, list the name of the party responsible for the registration. Contact

information such as mailing address, email address, or phone number may also be provided.

Defining Document(s)
A reference to the document or documents that define the Subject Identifier Type. The definition MUST specify the name, format, and meaning of each member that may occur within a Subject Identifier of the defined type, as well as whether each member is optional, required, prohibited, or the circumstances under which the member may be optional, required, or prohibited. URIs that can be used to retrieve copies of each document SHOULD be included.

7.1.3. Initial Registry Contents

7.1.3.1. Account Subject Identifier Type

- Type Name: "account"
- Type Description: Subject identifier based on "acct" URI.
- Change Controller: IETF
- Defining Document(s): Section 3 of this document.

7.1.3.2. Email Subject Identifier Type

- Type Name: "email"
- Type Description: Subject identifier based on email address.
- Change Controller: IETF
- Defining Document(s): Section 3 of this document.

7.1.3.3. Issuer and Subject Subject Identifier Type

- Type Name: "iss_sub"
- Type Description: Subject identifier based on an issuer and subject.
- Change Controller: IETF
- Defining Document(s): Section 3 of this document.

7.1.3.4. Phone Number Subject Identifier Type

- Type Name: "phone_number"
- Type Description: Subject identifier based on an phone number.
- Change Controller: IETF
- Defining Document(s): Section 3 of this document.

7.1.3.5. Aliases Subject Identifier Type

- Type Name: "aliases"
- Type Description: Subject identifier that groups together multiple different subject identifiers for the same subject.
- Change Controller: IETF
- Defining Document(s): Section 3 of this document.

7.1.4. Guidance for Expert Reviewers

The Expert Reviewer is expected to review the documentation referenced in a registration request to verify its completeness. The Expert Reviewer must base their decision to accept or reject the request on a fair and impartial assessment of the request. If the Expert Reviewer has a conflict of interest, such as being an author of a defining document referenced by the request, they must recuse themselves from the approval process for that request. In the case where a request is rejected, the Expert Reviewer should provide the requesting party with a written statement expressing the reason for rejection, and be prepared to cite any sources of information that went into that decision.

Subject Identifier Types need not be generally applicable and may be highly specific to a particular domain; it is expected that types may be registered for niche or industry-specific use cases. The Expert Reviewer should focus on whether the type is thoroughly documented, and whether its registration will promote or harm interoperability. In most cases, the Expert Reviewer should not approve a request if the registration would contribute to confusion, or amount to a synonym for an existing type.

7.2. JSON Web Token Claims Registration

This document defines the "sub_id" JWT Claim, which IANA is asked to register in the "JSON Web Token Claims" registry IANA JSON Web Token Claims Registry [IANA.JWT.Claims] established by [RFC7519].

7.2.1. Registry Contents

- Claim Name: "sub_id"
- Claim Description: Subject Identifier
- Change Controller: IESG
- Specification Document(s): Section 4.1 of this document.

8. References

8.1. Normative References

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[E164] International Telecommunication Union, "The international public telecommunication numbering plan", 2010, <http://www.itu.int/rec/T-REC-E.164-201011-I/en>.

[IANA.JWT.Claims] IANA, "JSON Web Token Claims", n.d., <http://www.iana.org/assignments/jwt>.

[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>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[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>.

[RFC7565] Saint-Andre, P., "The 'acct' URI Scheme", RFC 7565, DOI 10.17487/RFC7565, May 2015, <https://www.rfc-editor.org/info/rfc7565>.

[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>.

8.2. Informative References

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[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>.

Acknowledgements

The authors would like to thank the members of the IETF Security Events working group, as well as those of the OpenID Shared Signals and Events Working Group, whose work provided the original basis for this document.

Change Log

(This section to be removed by the RFC Editor before publication as an RFC.)

Draft 00 - AB - First draft

Draft 01 - AB:

- Added reference to RFC 5322 for format of "email" claim.
- Renamed "iss_sub" type to "iss-sub".
- Renamed "id_token_claims" type to "id-token-claims".
- Added text specifying the nature of the subjects described by each type.

Draft 02 - AB:

- Corrected format of phone numbers in examples.
- Updated author info.

Draft 03 - AB:

- Added "account" type for "acct" URIs.
- Replaced "id-token-claims" type with "aliases" type.
- Added email canonicalization guidance.
- Updated semantics for "email", "phone", and "iss-sub" types.

Draft 04 - AB:

- Added "sub_id" JWT Claim definition, guidance, examples.
- Added text prohibiting "aliases" nesting.
- Added privacy considerations for identifier correlation.

Draft 05 - AB:

- Renamed the "phone" type to "phone-number" and its "phone" claim to "phone_number".

Draft 06 - AB:

- Replaced usage of the word "claim" to describe members of a Subject Identifier with the word "member", in accordance with terminology in RFC7159.

- Renamed the "phone-number" type to "phone_number" and "iss-sub" to "iss_sub".
- Added normative requirements limiting the use of both "sub" and "sub_id" claims together when processing a JWT.
- Clarified that identifier correlation may be acceptable when it is a core part of the use case.
- Replaced references to OIDF with IETF in IANA Considerations.
- Recommended the appointment of multiple Designated Experts, and a location for the Subject Identifier Types registry.
- Added "_" to list of allowed characters in the Type Name for Subject Identifier Types.
- Clarified that Subject Identifiers don't provide confidentiality or integrity protection.
- Added references to SET, JWT privacy and security considerations.
- Added section describing the difference between subject identifier type and principal type that hopefully clarifies things and doesn't just muddy the water further.

Authors' Addresses

Annabelle Backman (editor)
Amazon

Email: richanna@amazon.com

Marius Scurtescu
Coinbase

Email: marius.scurtescu@coinbase.com