Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2017

P. Hunt, Ed.
Oracle
M. Scurtescu
Google

March 8, 2017

SET Token Delivery Using HTTP
draft-hunt-secevent-distribution-01

Abstract

This specification defines how a series of security event tokens (SETs) may be delivered to a previously registered receiver using HTTP over TLS. The specification defines the metadata the an Event Transmitter uses to describe the Event Receiver's HTTP endpoint and the SET token delivery configuration. The specification defines how the Event Receiver may check the current configuration metadata and delivery status using HTTP GET over TLS. The specification also defines how delivery can be assured subject to the SET Token Receiver'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 http://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 September 9, 2017.

Copyright Notice

Table of Contents

1. Introduction and Overview

This specification defines how a stream of SETs (see [I-D.ietf-secevent-token]) can be transmitted to a previously registered Event Receiver using HTTP POST [RFC7231] over TLS. The specification defines the metadata the Event Transmitter uses to describe the Event Receiver's HTTP endpoint and the SET token delivery configuration. The specification defines how the Event Receiver may check the current configuration metadata and delivery status using HTTP GET over TLS. The specification also defines how delivery can be assured subject to the SET Token Receiver's need for assurance.

The following diagram shows a typical SET Event Stream. A stream consists of a pair of HTTP endpoints, one for the event stream transmitter and one for the receiver. The receiver endpoint is used by the transmitter to deliver SET events via HTTPS POST and is known as the "Data Plane". The transmitter's HTTP endpoint is used by the receiver to perform HTTPS GET requests to check the stream status and is known as the "Control Plane". In the diagram, the arrow heads point to the service provider (the direction of an HTTP request):

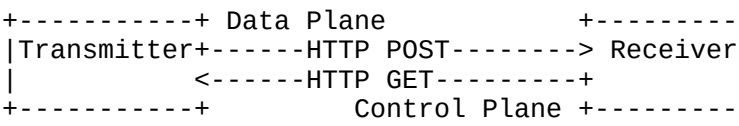

Figure 1: SET Event Stream

In some service provider relationships, for example between Identity Providers and Relying Parties, there may be a need to have bi-directional SET event exchange. This involves establishing a second event stream that works with transmitter and receiver roles reversed.

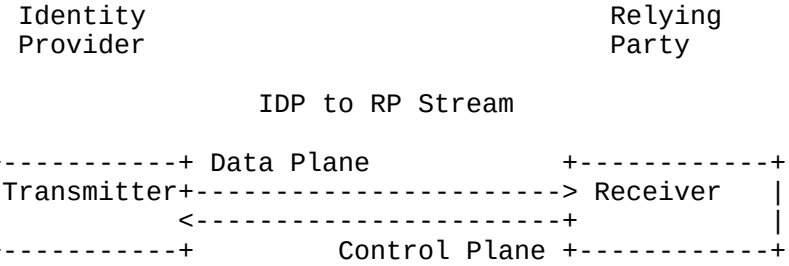

Figure 2: Duplexed Streams

This specification contains two major sections:

Control Plane The service through which Event Receivers can review and optionally managed Event Streams. It defines the metadata associated with Event Streams along with stream status reporting.

Data Plane Through which SET Events are delivered by an Event Transmitter to an Event Receiver using a defined Event Stream. The Data Plane includes a verification process which tests and validates Event Stream configuration. The Data plan defines processing and error signaling used in the delivery of SETs.

1.1. 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] . These keywords are capitalized when used to unambiguously specify requirements of the protocol or application features and behavior that affect the inter-operability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

For purposes of readability examples are not URL encoded. Implementers MUST percent encode URLs as described in Section 2.1 of [RFC3986] .

Throughout this documents all figures MAY contain spaces and extra line-wrapping for readability and space limitations. Similarly, some URI’s contained within examples, have been shortened for space and readability reasons.

1.2. Definitions

This specification assumes terminology defined in the Security Event Token specification[I-D.ietf-secevent-token] .

The following definitions are defined for Security Event distribution:

Identity Provider
An Identity Provider is a service provider that issues authentication assertions that may be used by Relying Party service providers to establish login sessions with users. Examples of Identity Providers are defined in: OpenID Connect [openid-connect-core] and SAML2 [saml-core-2.0]. For the purpose of this specification an Identity Provider also includes any provider of services where the compromise of an account may open up relying parties to attack. For example for the purposes of security events, an email service provider could be considered an "implicit" Identity Provider.

Relying Party

A Relying Party is a service provider that accepts assertions from Identity Providers to establish sessions. Examples of Relying Parties are defined in: OpenID Connect [openid-connect-core] and SAML2 [saml-core-2.0]

Event Transmitter
A service provider that delivers SETs to other providers known as Event Receivers. Some examples of Event Transmitters are Identity Providers and Relying Parties. An Event Transmitter is responsible for offering a service that allows the Event Receiver to check the Event Stream configuration and status known as the "Control Plane".

Event Receiver
A service provider that registers to receive SETs from an Event Transmitter and provides an endpoint to receive SETs via HTTP POST (known as the "Data Plane"). Some examples of Event Receivers are Identity Providers and Relying Parties. Event Receivers can check current Event Stream configuration and status by accessing the Event Transmitters "Control Plane".

Event Stream
An Event Stream establishes Event Receiver communication endpoints, security configuration and feed content that is used by an Event Transmitter to send a series of SET Events to an Event Receiver. An Event Stream defines a "Data Plane" and "Control Plane" service relationship between an Event Transmitter and and Event Receiver.

Control Plane
A Control Plane represents an service offered by an Event Transmitter that lets an Event Receiver query the current operational and/or error status of an Event Stream. The Control Plane MAY also be used to retrieve Event Stream and SET configuration data.

Data Plane
The Data Plane represents the HTTP service offered by an Event Receiver that allows the Event Transmitter to deliver multiple SETs via HTTP POST as part of an Event Stream.

Event Family
An Event Family is a URI that describes the set of events types be issued in an Event Stream.

Subject
The security subject around which a security event has occurred. For example, a security subject might per a user, a person, an

email address, a service provider entity, an IP address, an OAuth Client, a mobile device, or any identifiable thing referenced in security and authorization systems.

2. Control Plane - Monitoring

The Control Plane is provided by the Event Transmitter and enables Event Receivers to check the Event Stream configuration and check for transmission errors. This section describes mandatory to implement functionality to enable Event Receivers to detect SET delivery problems that may occur when an Event Transmitter fails to deliver SETs.

Implementers MAY optionally implement and support full Event Stream provisioning and management as described in Section 4. This functionality also allows Event Receivers to "pause", "disable", or re-enable Event Streams in scenario where the operational needs of the receiver need to be co-ordinated with Event Transmitters (see Section 2.2 and Section 4.3).

SCIM defines flexible mechanisms to ease adaptability to different underlying data systems while maximizing inter-operabilty. Section 2 [RFC7643] SHALL provide the processing rule that enable Control Plane providers and clients negotiate specific attributes (metadata) including differing provider definitions of attribute types, mutability, cardinality, or returnability that MAY differ. For HTTP method handling and error signaling, the processing rules in [RFC7644] SHALL apply.

2.1. Event Stream Configuration

An Event Stream represents an agreement to deliver SETs from a specified Feed URI from an Event Transmitter to an Event Receiver. The method of delivery and the parameters for delivery are specified a set of parameters called Event Stream metadata (see Section 2.1).

An Event Stream is defined by the following metadata:

feedUri
: An OPTIONAL JSON String value containing the URI for a feed supported by the feed provider. It describes the content of the feed and MAY also be a resolvable URI where the feed meta data may be returned as a JSON object. REQUIRED.

methodUri
: A REQUIRED JSON String value which is a URI with a prefix of "urn:ietf:params:set:method". This specification defines HTTP

POST delivery method:
"urn:ietf:params:set:method:HTTP:webCallback"
in which the Feed Provider delivers events using HTTP POST to a specified callback URI.

deliveryUri
A JSON String value containing a URI that describes the location where SETs are received (e.g. via HTTP POST). Its format and usage requirements are defined by the associated "methodUri".

aud
An OPTIONAL JSON Array of JSON String values which are URIs representing the audience(s) of the Event Stream. The value SHALL be the value of SET "aud" claim sent to the Event Receiver.

feedJwk
An OPTIONAL public JSON Web Key (see [RFC7517]) from the Event Transmitter that will be used by the Event Receiver to verify the authenticity of issued SETs.

confidentialJwk
An OPTIONAL public JSON Web Key (see [RFC7517]) for the Event Receiver that MAY be used by the Feed Provider to encrypt SET tokens for the specified Event Receiver.

subStatus
An OPTIONAL JSON String keyword that indicates the current state of an Event Stream. More information on the Event Stream state can be found in Section 2.2. Valid keywords are:

"on" - indicates the Event Stream has been verified and that the Feed Provider MAY pass SETs to the Event Receiver.

"verify" - indicates the Event Stream is pending verification. While in "verify", SETs, except for the verify SET (see Section 3.4) are not delivered to the Event Receiver. Once verified, the status returns to "on".

"paused" - indicates the Event Stream is temporarily suspended. While "paused", SETs SHOULD be retained and delivered when state returns to "on". If delivery is paused for an extended period defined by the Event Transmitter, the Event Transmitter MAY change the state to "off" indicating SETs are no longer retained.

"off" - indicates that the Event Stream is no longer passing SETs. While in off mode, the Event Stream metadata is maintained, but new events are ignored, not delivered or

retained. Before returning to "on", a verification MUST be performed.

"fail" - indicates that the Event Stream was unable to deliver SETs to the Event Receiver due an unrecoverable error or for an extended period of time. Unlike paused status, a failed Event Stream does not retain existing or new SETs that are issued. Before returning to "on", a verification MUST be performed.

maxRetries
An OPTIONAL JSON number indicating the maximum number of attempts to deliver a SET. A value of '0' indicates there is no maximum. Upon reaching the maximum, the Event Stream "subStatus" attribute is set to "failed".

maxDeliveryTime
An OPTIONAL number indicating the maximum amount of time in seconds a SET MAY take for successful delivery per request or cumulatively across multiple retries. Upon reaching the maximum, the Event Stream "subStatus" is set to "failed". If undefined, there is no maximum time.

minDeliveryInterval
An OPTIONAL JSON integer that represents the minimum interval in seconds between deliveries. A value of '0' indicates delivery should happen immediately. When delivery is a polling method (e.g. HTTP GET), it is the expected time between Event Receiver attempts. When in push mode (e.g. HTTP POST), it is the interval the server will wait before sending a new event or events.

txErr
An OPTIONAL JSON String keyword value. When the Event Stream has "subState" set to "fail", one of the following error keywords is set:

"connection" indicates an error occurred attempting to open a TCP connection with the assigned endpoint.

"tls" indicates an error occurred establishing a TLS connection with the assigned endpoint.

"dnsname" indicates an error occurred establishing a TLS connection where the dnsname was not validated.

"receiver" indicates an error occurred whereby the Event Receiver has indicated an error for which the Event Transmitter is unable to correct.

[[Editors note: other conditions?]]

txErrDesc
An OPTIONAL String value that is usually human readable that provides further diagnostic detail by the indicated "txErr" error code.

Additional Event Stream metadata (attributes) MAY be defined as extensions. The method for adding new attributes is defined in Section 3.3 [RFC7643].

2.2. Event Stream State Model

The Event Stream configuration attribute "subStatus" tracks the state of any particular Event Stream with regards to whether SETs are ready or able to be delivered. The impact on delivery processing is described in Table 1.

The following is the state machine representation of a Event Stream on a Event Transmitter. Note that a Event Stream cannot be made active until a verification process has been completed. As such, a newly created Event Stream begins with state "verify".

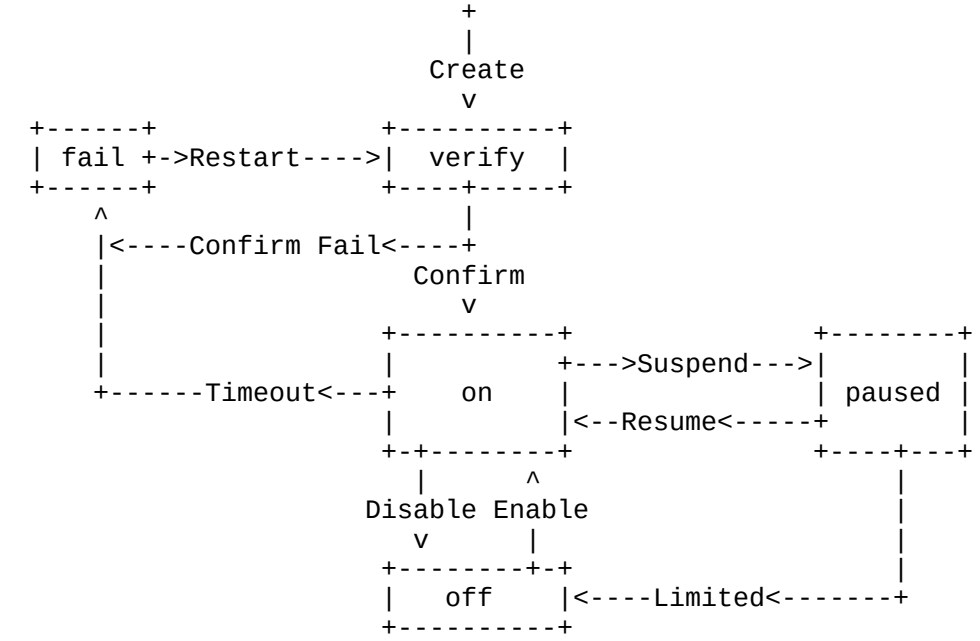

Figure 3: Event Stream States at Event Transmitter

In the above diagram, the following actions impact the state of an Event Stream. "subStatus" values are shown in the boxes, and change based on the following actions:

Create
: A Event Receiver or an administrator creates a new Event Stream using SCIM as described in Section 4.2. The initial state is "verify".

Confirm
: The Event Transmitter sends a verification SET to the Event Receiver which confirms with the correct response as described in Section 3.4. If it succeeds to deliver, the Event Transmitter SHALL set state to "on".

Confirm Fail
: If the confirmation fails, the Event Transmitter sets the state to "fail" requiring administrative action to correct the issue and "Restart".

Timeout
: A Event Transmitter who has not been able to deliver a SET over one or more retries which has reached a limit of attempts ("maxRetries") or time ("maxDeliveryTime") MAY set the Event Stream state to "fail". In general, the intention is to indicate the maximum number of retries or time a Event Transmitter is able to wait until SET event loss begins to occur resulting in the failed state.

Limited
: A paused Event Stream has reached a limit and the Event Transmitter can no longer retain SETs. The Event Transmitter changes the state to "off".

Restart
: An administrator having corrected the failed delivery condition modifies the Event Stream state to "verify" (e.g. see Section 4.3).

Suspend and Resume
: An Event Stream MAY be suspended and resumed by updating the Event Stream state to "paused" or "on". For example, see see Section 4.3. While suspended, the Event Transmitter MAY retain undelivered SETs for a period of time. If the Event Transmitter is no longer able to retain SETs, the Event Stream state SHOULD be set to "off" to indicate SETs are being lost.

Enable and Disable

A Event Stream MAY be disabled and enabled by updating the Event Stream state to "off" or "on". For example, see see Section 4.3. While the Event Stream is disabled, all SETs that occur at the Event Transmitter are lost.

2.3. Checking Stream Configuration and Stream State

An Event Receiver MAY check the current status of a Stream with the Event Transmitter, by performing an HTTP GET using the provided URI from the Transmitter.

The format of the response is defined by Section TBD [RFC7644].

In addition to the attributes defined in Section 2.1, the response SHALL include an additional JSON attribute "schemas" with at least a single value of "urn:ietf:params:scim:schemas:event:2.0:EventStream". This static attribute is provided to enable optional SCIM client compatibility and informs the client of the type of JSON object being returned. Service providers may offer additional attributes by adding additional schema values as per [RFC7644].

The response below shows an example response to an HTTP GET, in this case to "https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0".

```
HTTP/1.1 200 OK
Content-Type: application/json
Location:
 https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "subStatus":"fail",
  "txErr":"connection",
  "txErrDesc":"TCP connect error to notify.examplerp.com.",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Figure 4: Example Stream GET Response

In the above figure, the Event Stream is showing a failed status due to a TCP connection error. The Event Receiver is able to discover that its endpoint was unavailable and has been marked failed by the Event Transmitter. It is expected that the appropriate operations staff would be alerted and some corrective action would be taken.

The frequency with which Event Receivers should poll the Event Stream status depends on the following factors:

- The level of technical fault tolerance and availability of the receiving endpoint.
- A frequency appropriate to the amount of risk that can be tolerated for lost events. For example, if Security Events are considered informational, then infrequent (hourly or daily) may be sufficient.

In most cases Event Stream status polling can be triggered on a timeout basis. Event Receivers would typically poll if they have not received a SET for some period during which SETs would be expected based on past experience.

3. Data Plane

The data plane represent the HTTP request channel by which the Event Transmitter delivers SET Events to an Event Receiver.

3.1. Event Delivery Process

When a Security Event occurs, the Feed Provider constructs a SET token [I-D.ietf-secevent-token] that describes the event. The feed provider determines the feeds that the event should be distributed to, and determines which Event Receivers need to be notified.

How SET Events are defined and the process by which events are identified for Event Receivers is out-of-scope of this specification.

When a SET is available for a Event Receiver, the Feed Transmitter attempts to deliver the SET based on the Event Receiver's registered delivery mechanism:

- The Event Transmitter uses an HTTP/1.1 POST to the Event Receiver endpoint to deliver the SET;
- Or, the Feed Transmitter delivers the event through a different method not defined by this specification.

Feed Transmitters SHALL NOT be required to main or record SETs. As such, transmitted SETs SHOULD be self-validating (e.g. signed).

If delivery to any particular Event Receiver has been delayed for an extended period of time, the Feed Transmitter MAY suspend the affected Event Stream and even stop maintaining outstanding SETs for the Event Receiver at its discretion and available resources. See Event Stream "subState" in Section 2.1.

Upon receiving a SET, the Event Receiver reads the SET and validates it. Based upon the content of the token, the Event Receiver decides what, if any, action needs to be taken in response to the received SET. For example, in response to a SCIM provisioning event [idevent-scim] indicating a changed resource, the Event Receiver might perform a SCIM GET request (see Section 3.4 [RFC7644]) to the affected resource URI in order to confidentially obtain the current state of the transmitter's affected SCIM resource in order to reconcile local corresponding state changes.

The action a Event Receiver takes in response to a SET MAY be substantially different than merely copying the action of the SET issuer. A single SET can trigger one or more receiver actions or it can be ignored. For example, upon receiving notification that a user resource has been added to a group, the Event Receiver may first determine that the user does not exist in the Event Receiver's domain. The Event Receiver translates the event into two actions:

1. Retrieve the user (e.g. using SCIM GET) and then provisions the user locally. After enabling the user,

2. The Event Receiver then enables the user for the application associated with membership in the issuer's group.

3.2. Event Stream State

As mentioned in Section 2.1, the attribute "subStatus" defines the current state of an Event Stream. Figure 3 shows a state diagram for Event Streams. The following describes that actions taken by the Event Transmitter based upon "subStatus".

Status	Action
on	Delivery SHALL be attempted based on the method defined in the Event Stream attribute "methodUri". If the SET fails to deliver it MAY be retained for a retry delivery in a minimum of "minDeliveryInterval" seconds. If new SETs arrive before the interval, the SETs MUST be held for delivery in order of reception. If this is a repeat attempt to deliver, the Event Transmitter MAY discard the SET if "maxRetries" or "maxDeliveryTime" is exceeded. If a SET is discarded, the Event Transmitter MAY set "subStatus" to "failed".
verify	If the SET is not a Verify SET, the SET MAY be retained for a retry at the Event Transmitter's discretion. If a Verify SET fails to deliver, the Event Transmitter SHALL set "subStatus" to "failed". The Event Transmitter MAY opt to make multiple attempts to complete a verification during which status remains as "verify".
paused	The SET is held for delivery in a queue. The Event Transmitter MAY at its own discretion set the Event Stream state to "failed" if "subStatus" is not returned to "on" in what the Event Transmitter determines to be a reasonable amount of time.
off	The SET is ignored.
fail	The SET is ignored due to a previous unrecoverable error.

Table 1: Delivery Processing By Status

3.3. HTTP POST Delivery

This method allows a feed provider to use HTTP POST (Section 4.3.3 [RFC7231]) to deliver SETs to the registered web callback URI identified in the Event Stream configuration. The Event Stream "methodUri" value for this method is "urn:ietf:params:set:method:HTTP:webCallback".

The SET to be delivered MAY be signed and/or encrypted as defined in [I-D.ietf-secevent-token].

The Event Stream's "deliveryUri" attribute indicates the location of a Event Receiver provided endpoint which accepts HTTP POST requests (e.g. "https://notify.examplerp.com/Events").

The content-type for the HTTP POST is "application/jwt" and SHALL consists of a single SET token (see [I-D.ietf-secevent-token]).

```
eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6Indvcmsi LCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
.
```

Figure 5: Encoded SET To Be Transmitted

To deliver an event, the Event Transmitter generates an event delivery message and uses HTTP POST to the EventStream configured endpoint. The content-type of the message is "application/jwt" and the expected response type (accept) is "application/json".

```
POST /Events  HTTP/1.1

Host: notify.examplerp.com
Accept: application/json
Content-Type: application/jwt
"eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6Indvcmsi LCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
.
```

Figure 6: Example Web Callback POST Request

Upon receipt of the request, the Event Receiver SHALL validate the JWT structure of the SET as defined in Section 7.2 [RFC7519]. The Event Receiver SHALL also validate the SET information as described in Section 2 [I-D.ietf-secevent-token].

If the SET is determined to be valid, the Event Receiver SHALL indicate successful submission by responding with HTTP Status 202 as "Accepted" (see Section 6.3.3 [RFC7231]).

If SET or JWT is invalid, or there is an HTTP error, the Event Receiver SHALL respond with the appropriate HTTP error or an HTTP Status 400 Bad Request error as follows:

Err Value	Description
jwtParse	Invalid or unparsable JWT or JSON structure.
jwtHdr	In invalid JWT header was detected.
jwtCypto	Unable to parse due to unsupported algorithm.
jws	Signature was not validated.
jwe	Unable to decrypt JWE encoded data.
jwtAud	Invalid audience value.
jwtIss	Issuer not recognized.
setType	An unexpected event type was received.
setParse	Invalid structure was encountered such as inability to parse SET event payload.
setData	SET event claims incomplete or invalid.
dup	A duplicate SET was received and has been ignored.

Table 2: HTTP Status 400 Errors

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

```
HTTP/1.1 202 Accepted
```

Figure 7: Example Successful Delivery Response

An HTTP Status 400 Bad Request response includes a JSON object which provides details about the error. The JSON object includes the JSON attributes:

err
: A value which is a keyword that describes the error (see Table 2).

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

The following is an example non-normative Bad Request error.

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

{
  "err":"dup",
  "description":"SET already received. Ignored."

}
```

Figure 8: Example Bad Request Response

3.4. Event Stream Verification

To confirm an Event Stream configuration, the Event Transmitter SHALL send a verification SET to the Event Receiver using the registered "methodUri" mechanism which in this case is "urn:ietf:params:set:method:HTTP:webCallback".

The Verify SET contains the following attributes:

events Set with a value of "[[this RFC URL]]#verify".

iss Set to the URI defined in the Event Stream metadata (see Section 2.1).

aud MUST be set to a value that matches the EventStream "aud" value (see Section 2.1).

exp A value that indicates the time the verification request will expire. Once expired, the server will set the Event Stream state to "fail".

If the Event Stream "confidentialJWK" value was supplied, then the SET SHOULD be encrypted with the provided key. Successful parsing of the message confirms that provides confirmation of correct configuration and possession of keys.

A payload attribute "confirmChallenge" is provided with a JSON String value that the Event Receiver SHALL echo back in its response. The intent is to confirm that the Event Receiver has successfully parsed the SET and is not just echoing back HTTP success.

A non-normative JSON representation of an event to be sent to a Event Receiver as a Event Stream confirmation. Note the event is not yet encoded as a JWT token:

```
{
  "jti": "4d3559ec67504aaba65d40b0363faad8",
  "events":["[[this RFC URL]]#verify"],
  "iat": 1458496404,
  "iss": "https://scim.example.com",
  "exp": 1458497000,
  "aud":[
   "https://scim.example.com/Feeds/98d52461fa5bbc879593b7754",
   "https://scim.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],
  "[[this RFC URL]]#verify":{
    "confirmChallenge":"ca2179f4-8936-479a-a76d-5486e2baacd7"
  }
}
```

Figure 9: Example Verification SET with Challenge

The above SET is encoded as a JWT and transmitted to the Event Receiver as shown in Figure 6.

Upon receiving a verify SET, the Event Receiver SHALL respond with a JSON object that includes a "challengeResponse" attribute and the value that was provided in "confirmChallenge". The content type header is set to "application/json".

The following is a non-normative example response to a Verify SET received via HTTP/1.1 POST and includes a JSON object containing the confirmation attribute and value.

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

{
  "challengeResponse":"ca2179f4-8936-479a-a76d-5486e2baacd7"
}
```

Figure 10: Example Response to Verify SET with Challenge

If the Event Receiver returns a non-matching value or an HTTP status other than a 200 series response, the Event Stream "state" SHALL be set to "fail". A declining Event Receiver MAY simply respond with any 400 series HTTP error (e.g. 404).

4. Control Plane - Management and Provisioning

This section describes how SCIM [RFC7644] and [RFC7643] MAY be used to add create, read, update, delete capability to the Control Plane to enable provisioning and operational management of Event Streams. In addition to provisioning of Event Streams, it can also be used by Event Receivers to change or reset the operational state of Event Streams such as pausing, stopping, or re-enabling after a failure.

SCIM is a protocol used by many security systems for provisioning and co-ordinating identities and other security subjects in cross-domain scenarios. SCIM is a RESTful profile of HTTP that is intended to be implemented by applications that need provisioning and management of security subjects and is ideal to the task of provisioning related security event signal systems. Examples of provisioning endpoints (SCIM service providers) include both Identity Providers and Relying Party applications (e.g. business and consumer web applications) as well as security and authorization infrastructure components.

[[Editors Note: At the time of writing, some groups feel a CRUD API is not required and participants would prefer to manage streams using an out-of-band workflow approach.]]

4.1. Event Stream Resource Type Definition

To extend SCIM to support Event Streams, requires defining an "EventStream" SCIM resource type, and implementing the corresponding RESTful HTTP operations to create, update, retrieve EventStream Resources. For SCIM service provider capability and schema discovery (see Sections 3 and 4 [RFC7644]).

The "EventStream" resource type definition is defined as follows:

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
  "id": "EventStream",
  "name": "EventStream",
  "endpoint": "/EventStreams",
  "description": "Endpoint and event configuration and status for SEC EVENT stre
ams.",
  "schema": "urn:ietf:params:scim:schemas:event:2.0:EventStream",
  "schemaExtensions": []
}
```

The resource type above is discoverable in the "/ResourceTypes" and informs SCIM clients about the endpoint location of EventStream resources and the SCIM schema used to define the resource. The corresponding schema for the EventStream resource MAY be retrieved from the SCIM "/Schemas" endpoint (see Section 3.2 [RFC7644]).

Figure 11: SCIM EventStream Resource Type Definition

To retrieve information about one or more Event Streams, authorized clients MAY query the "/EventStreams" endpoint as defined in Section 3.4 [RFC7644].

The example below retrieves a specific "EventStream" resource whose "id" is "548b7c3f77c8bab33a4fef40".

```
GET /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

Figure 12: Example SCIM EventStream HTTP GET Request

The response below shows an example Feed resource that describes an available feed.

```
HTTP/1.1 200 OK
Content-Type: application/json
Location:
 https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "subStatus":"verify",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Figure 13: Example EventStream HTTP GET Response

In the above example (Figure 13) the EventStream is for the the Feed "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74". The current Event Stream state is "verify" which suggest the Event Stream Verification (see Section 3.4) process has not yet completed. Since there is no value for "feedJwk",) or "confidentialJwk", SETs will be sent without signing or encryption (plain text).

4.2. Creating A New Event Stream

To subscribe to a feed, the Event Receiver first obtains an authorization credential authorizing to to make the request (this process is out of scope of the specification but is often completed through OAuth). Upon obtaining authorization, the Event Receiver organization uses the SCIM Create operation (HTTP POST) as defined in Section 3.3 [RFC7644]. Event Transmitter’s Control Plane service MAY have additional schema requirements for Event Stream creation which MAY be discovered using SCIM service configuration and schema discovery, see Section 4 [RFC7644].

In the following non-normative example, a new EventStream is created. Note that the Event Transmitter's control-plane automatically assigns the "id" attribute.

```
POST /EventStreams
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "feedName":"OIDCLogoutFeed",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com"
}
```

Figure 14: Example Create Event Stream Request

In following non-normative response, the Event service provider has automatically assigned a resource location as well as an "id". Usually upon creation, the initial value of "subStatus" is "pending" indicating that the Stream Verification process (see Section 3.4) has not been completed.

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
 https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "subStatus":"verify",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Figure 15: Example Response to Create EventStream Request

4.3. Updating An Event Stream

Periodically, Event Receivers MAY have need to update an Event Stream configuration for the purpose of:

- Rotating access credentials or keys
- Updating endpoint configuration
- Making operational changes such as pausing, resetting, or disabling an Event Stream.
- Other operations (e.g. such as adding or removing subjects) as defined by profiling Event specifications.

To modify an EventStream, an Event Receiver or authorized management client MAY use the HTTP PUT operation (see Section 3.5.1 [RFC7644])

or MAY use the HTTP PATCH operation (see Section 3.5.2 [RFC7644]) if supported by the Event Transmitter's control plane service. Note that HTTP PATCH enables more specific changes. This is particularly useful when updating multi-valued attributes that may contain large numbers of values. An example of this would be an EventStream that uses a "members" attribute to define the subjects of the Event Stream.

In the following non-normative example, the client is requesting that "subStatus" be changed to "paused" for the EventStream whose path is identified by the request URI path.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations": [{
    "op":"replace",
    "path":"subStatus",
    "value":"paused"
  }]
}
```

Upon receiving the request, the Event Transmitter would stop sending Events to the Receiver. Note that while the request MAY seem complex it avoids the need for the requestor to have all of the current EventStream values in order to make a PUT request. In other words, an HTTP PATCH can be typically done in a single request response whereas an HTTP POST usually is preceded by an HTTP GET.

Figure 16: Example EventStream PATCH Request

5. Security Considerations

[TO BE COMPLETED]

6. IANA Considerations

6.1. SCIM Schema Registration

As per the "SCIM Schema URIs for Data Resources" registry established by Section 10.3 [RFC7643], the following defines and registers the following SCIM URIs and Resource Types for Feeds and Event Streams.

Schema URI	Name	ResourceType	Reference
urn:ietf:params:scim: schemas:event:2.0: EventStream	SET Event Stream	EventStream	Section 2.1

7. References

7.1. Normative References

[I-D.ietf-secevent-token]
Hunt, P., Denniss, W., Ansari, M., and M. Jones, "Security Event Token (SET)", draft-ietf-secevent-token-00 (work in progress), January 2017.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[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, <http://www.rfc-editor.org/info/rfc7231>.

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

7.2. Informative References

[idevent-scim]
Oracle Corporation, "SCIM Event Extensions (work in progress)".

[openid-connect-core]
NRI, "OpenID Connect Core 1.0", Nov 2014.

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

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

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

[RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, DOI 10.17487/RFC7643, September 2015, <http://www.rfc-editor.org/info/rfc7643>.

[RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <http://www.rfc-editor.org/info/rfc7644>.

[saml-core-2.0]
Internet2, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005.

Appendix A. Acknowledgments

The editors would like to thanks the members of the SCIM WG which began discussions of provisioning events starting with: draft-hunt-scim-notify-00 in 2015.

The editor would like to thank the participants in the the SECEVENTS working group for their support of this specification.

Appendix B. Change Log

Draft 00 - PH - First Draft based on reduced version of draft-hunt-idevent-distribution

Draft 01 - PH -

- Reworked terminology to match new WG Transmitter/Receiver terms
- Reworked sections into Data Plane vs. Control Plane
- Removed method transmission registry in order to simplify the specification
- Made Create, Update operations optional for Control Plane (Read is MTI)

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Marius Scurtescu
Google

Email: mscurtescu@google.com

Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2018

P. Hunt, Ed.
Oracle
M. Jones
Microsoft
W. Denniss
Google
M. Ansari
Cisco

March 4, 2018

Security Event Token (SET)
draft-ietf-secevent-token-07

Abstract

This specification defines the Security Event Token (SET) data structure. A SET describes a statement of fact from the perspective of an issuer about the state of a security subject, which is intended to be shared with one or more recipients. This statement of fact represents an event that occurred to the security subject. In some use cases, the security subject may be a digitial identity, but SETs are also applicable to non-identity use cases. A SET is a JSON Web Token (JWT), which can be optionally signed and/or encrypted. SETs can be distributed via protocols such as HTTP.

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 September 5, 2018.

Table of Contents

1. Introduction and Overview

This specification defines an extensible Security Event Token (SET) data structure, which can be exchanged using protocols such as HTTP. The specification builds on the JSON Web Token (JWT) format [RFC7519] in order to provide a self-contained token that can be optionally signed using JSON Web Signature (JWS) [RFC7515] and/or encrypted using JSON Web Encryption (JWE) [RFC7516].

This specification profiles the use of JWT for the purpose of issuing Security Event Tokens (SETs). This specification defines a base format used by profiling specifications to define actual events and their meanings. This specification uses non-normative example events to demonstrate how events can be constructed.

This specification is scoped to security and identity related events. While Security Event Tokens may be used for other purposes, the specification only considers security and privacy concerns relevant to identity and personal information.

Security events are not commands issued between parties. A security event is a statement of fact from the perspective of an issuer about the state of a security subject (e.g., a web resource, token, IP address, the issuer itself) that the issuer controls or is aware of, that has changed in some way (explicitly or implicitly). A security subject may be permanent (e.g., a user account) or temporary (e.g., an HTTP session) in nature. A state change could describe a direct change of entity state, an implicit change of state, or other higher-level security statements such as:

- The creation, modification, removal of a resource.
- The resetting or suspension of an account.
- The revocation of a security token prior to its expiry.
- The logout of a user session. Or,
- An indication that a user has been given control of an email identifier that was previously controlled by another user.

While subject state changes are often triggered by a user agent or security subsystem, the issuance and transmission of an event may occur asynchronously and in a back channel to the action that caused the change that generated the security event. Subsequently, a SET recipient, having received a SET, validates and interprets the received SET and takes its own independent actions, if any. For example, having been informed of a personal identifier being

associated with a different security subject (e.g., an email address is being used by someone else), the SET recipient may choose to ensure that the new user is not granted access to resources associated with the previous user. Or, the SET recipient may not have any relationship with the subject, and no action is taken.

While SET recipients will often take actions upon receiving SETs, security events cannot be assumed to be commands or requests. The intent of this specification is to define a syntax for statements of fact that SET recipients may interpret for their own purposes.

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.

For purposes of readability, examples are not URL encoded. Implementers MUST percent encode URLs as described in Section 2.1 of [RFC3986].

Throughout this document, all figures MAY contain spaces and extra line-wrapping for readability and space limitations. Similarly, some URIs contained within examples have been shortened for space and readability reasons.

1.2. Definitions

The following definitions are used with SETs:

Security Event Token (SET)
: A SET is a JWT [RFC7519] conforming to this specification that is distributed to one or more SET recipients.

SET Issuer
: A service provider that creates SETs to be sent to other service providers known as SET recipients.

SET Recipient
: A SET recipient is an entity that receives SETs through some distribution method. A SET recipient is the same entity referred as a "recipient" in [RFC7519] or "receiver" in related specifications.

Subject

A SET describes an event or state change that has occurred to a subject. A subject might, for instance, be a principal (e.g., Section 4.1.2 of [RFC7519]), a web resource, an entity such as an IP address, or the issuer of the SET.

Event Identifier
: A member name for an element of the JSON object that is the value of the "events" claim in a SET. This member name MUST be a URI.

Event Payload
: A member value for an element of the JSON object that is the value of the "events" claim in a SET. This member value MUST be JSON object.

Profiling Specification
: A specification that profiles the SET data structure to define one or more specific event types and their associated claims and processing rules.

2. The Security Event Token (SET)

A SET is a JWT [RFC7519] data structure that represents one or more related aspects of a security event that occurred to a subject. The JWT Claims Set in a SET has the following structure:

- The top-level claims in the JWT Claims Set are called the SET "envelope". Some of these claims are present in every SET; others will be specific to particular SET profiles or profile families. Claims in the envelope SHOULD be registered in the "JSON Web Token Claims" registry [IANA.JWT.Claims] or be Public Claims or Private Claims, as defined in [RFC7519].

- Envelope claims that are profiled and defined in this specification are used to validate the SET and provide information about the event data included in the SET. The claim "events" contains the event identifiers and event-specific data expressed about the security subject. The envelope MAY include event-specific or profile-specific data.

- Each member of the "events" JSON object is a name/value pair. The JSON member name is a URI string value, which is the event identifier, and the corresponding value is a JSON object known as the event "payload". The payload JSON object contains claims that pertain to that event identifier and need not be registered as JWT claims. These claims are defined by the profiling specification that defines the event. An event with no payload claims SHALL be represented as the empty JSON object ("{}").

- When multiple event identifiers are contained in a SET, they represent multiple aspects of the same state transition that occurred to the security subject. They are not intended to be used to aggregate distinct events about the same subject. Beyond this, the interpretation of SETs containing multiple event identifiers is out of scope for this specification; profiling specifications MAY define their own rules regarding their use of SETs containing multiple event identifiers, as described in Section 3. Possible uses of multiple values include, but are not limited to:
  - Values to provide classification information (e.g., threat type or level).
  - Additions to existing event representations.
  - Values used to link potential series of events.
  - Specific-purpose event URIs used between particular SET issuers and SET recipients.

2.1. Illustrative Examples

2.1.1. SCIM Example

The following is a non-normative example showing the JWT Claims Set for a hypothetical SCIM [RFC7644] password reset SET. This example uses a second "events" value ("https://example.com/scim/event/passwordResetExt") to convey additional information about the state change -- in this case, the current count of reset attempts:

```
{
  "iss": "https://scim.example.com",
  "iat": 1458496025,
  "jti": "3d0c3cf797584bd193bd0fb1bd4e7d30",
  "aud": [
    "https://jhub.example.com/Feeds/98d52461fa5bbc879593b7754",
    "https://jhub.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],
  "sub": "https://scim.example.com/Users/44f6142df96bd6ab61e7521d9",
  "events": {
    "urn:ietf:params:scim:event:passwordReset":
      { "id": "44f6142df96bd6ab61e7521d9"},
    "https://example.com/scim/event/passwordResetExt":
      { "resetAttempts": 5}
  }
}
```

Figure 1: Example SCIM Password Reset Event

The JWT Claims Set consists of:

- The "events" claim specifying the hypothetical SCIM URN ("urn:ietf:params:scim:event:passwordReset") for a password reset, and a second value, "https://example.com/scim/event/passwordResetExt", that is used to provide additional event information such as the current count of resets.

- The "iss" claim, denoting the SET issuer.

- The "sub" claim, specifying the SCIM resource URI that was affected.

- The "aud" claim, specifying the intended audiences for the event. (The syntax of the "aud" claim is defined in Section 4.1.3 of [RFC7519].)

In this example, the SCIM event indicates that a password has been updated and the current password reset count is 5. Notice that the value for "resetAttempts" is in the event payload of an event used to convey this information.

2.1.2. Logout Example

Here is another example JWT Claims Set for a security event token, this one for a Logout Token:

```
{
   "iss": "https://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "iat": 1471566154,
   "jti": "bWJq",
   "sid": "08a5019c-17e1-4977-8f42-65a12843ea02",
   "events": {
     "http://schemas.openid.net/event/backchannel-logout": {}
   }
}
```

Figure 2: Example OpenID Back-Channel Logout Event

Note that the above SET has an empty JSON object and uses the JWT registered claims "sub" and "sid" to identify the subject that was logged out.

2.1.3. Consent Example

In the following example JWT Claims Set, a fictional medical service collects consent for medical actions and notifies other parties. The individual for whom consent is identified was originally authenticated via OpenID Connect. In this case, the issuer of the security event is an application rather than the OpenID provider:

```
{
  "iss": "https://my.med.example.org",
  "iat": 1458496025,
  "jti": "fb4e75b5411e4e19b6c0fe87950f7749",
  "aud": [
    "https://rp.example.com"
  ],
  "events": {
    "https://openid.net/heart/specs/consent.html": {
      "iss": "https://connect.example.com",
      "sub": "248289761001",
      "consentUri": [
        "https://terms.med.example.org/labdisclosure.html#Agree"
      ]
    }
  }
}
```

Figure 3: Example Consent Event

In the above example, the attribute "iss" contained within the payload for the event "https://openid.net/heart/specs/consent.html" refers to the issuer of the security subject ("sub") rather than the SET issuer "https://my.med.example.org". They are distinct from the top-level value of "iss", which always refers to the issuer of the event -- a medical consent service that is a relying party to the OpenID Provider.

2.1.4. RISC Example

The following example JWT Claims Set is for an account disabled event. This example was taken from a working draft of the RISC events specification, where RISC is the OpenID RISC (Risk and Incident Sharing and Coordination) working group [RISC]. The example is subject to change.

```
{
  "iss": "https://idp.example.com/",
  "jti": "756E69717565206964656E746966696572",
  "iat": 1508184845,
  "aud": "636C69656E745F6964",
  "events": {
    "http://schemas.openid.net/secevent/risc/event-type/\
    account-disabled": {
      "subject": {
        "subject_type": "iss-sub",
        "iss": "https://idp.example.com/",
        "sub": "7375626A656374"
      },
      "reason": "hijacking",
      "cause-time": 1508012752
    }
  }
}
```

Figure 4: Example RISC Event

Notice that parameters to the event are included in the event payload, in this case, the "reason" and "cause-time" values. The subject of the event is identified using the "subject" payload value, which itself is a JSON object.

2.2. Core SET Claims

The following claims from [RFC7519] are profiled for use in SETs:

"iss" (Issuer) Claim
: As defined by Section 4.1.1 of [RFC7519], this claim contains a string identifying the service provider publishing the SET (the issuer). In some cases, the SET issuer is not the issuer of the security subject. Therefore, implementers cannot assume that the issuers are the same unless the profiling specification specifies that they are for SETs conforming to that profile. This claim is REQUIRED.

"iat" (Issued At) Claim

As defined by Section 4.1.6 of [RFC7519], this claim contains a value representing when the SET was issued. This claim is REQUIRED.

"jti" (JWT ID) Claim
As defined by Section 4.1.7 of [RFC7519], this claim contains a unique identifier for the SET. The identifier SHOULD be unique within a particular event feed and MAY be used by clients to track whether a particular SET has already been received. This claim is REQUIRED.

"aud" (Audience) Claim
As defined by Section 4.1.3 of [RFC7519], this claim contains one or more audience identifiers for the SET. This claim is RECOMMENDED.

"sub" (Subject) Claim
As defined by Section 4.1.2 of [RFC7519], this claim contains a StringOrURI value representing the principal that is the subject of the SET. This is usually the entity whose "state" was changed. For example, an IP Address was added to a black list. A URI representing a user resource that was modified. A token identifier for a revoked token. If used, the profiling specification SHOULD define the content and format semantics for the value. This claim is OPTIONAL, as the principal for any given profile may already be identified without the inclusion of a subject claim. Note that some SET profiles MAY choose to convey event subject information in the event payload (either using the "sub" member name or another name), particularly if the subject information is relative to issuer information that is also conveyed in the event payload, which may be the case for some identity SET profiles.

"exp" (Expiration Time) Claim
As defined by Section 4.1.4 of [RFC7519], this claim is the time after which the JWT MUST NOT be accepted for processing. In the context of a SET however, this notion does not typically apply, since a SET represents something that has already occurred and is historical in nature. Therefore, its use is NOT RECOMMENDED. (Also, see Section 4.5 for additional reasons not to use the "exp" claim in some SET use cases.)

The following new claims are defined by this specification:

"events" (Security Events) Claim
This claim contains a set of event statements that each provide information describing a single logical event that has occurred about a security subject (e.g., a state change to the subject).

Multiple event identifiers with the same value MUST NOT be used. The "events" claim SHOULD NOT be used to express multiple independent logical events.

The value of the "events" claim is a JSON object whose members are name/value pairs whose names are URIs identifying the event statements being expressed. Event identifiers SHOULD be stable values (e.g., a permanent URL for an event specification). For each name present, the corresponding value MUST be a JSON object. The JSON object MAY be an empty object ("{}"), or it MAY be a JSON object containing data described by the profiling specification.

"txn" (Transaction Identifier) Claim
An OPTIONAL string value that represents a unique transaction identifier. In cases in which multiple related JWTs are issued, the transaction identifier claim can be used to correlate these related JWTs. Note that this claim can be used in JWTs that are SETs and also in JWTs using non-SET profiles.

"toe" (Time of Event) Claim
A value that represents the date and time at which the event occurred. This value is a NumericDate (see Section 2 of [RFC7519]). By omitting this claim, the issuer indicates that they are not sharing an event time with the recipient. (Note that in some use cases, the represented time might be approximate.) This claim is OPTIONAL.

2.3. Explicit Typing of SETs

This specification registers the "application/secevent+jwt" media type, which can be used to indicate that the content is a SET. SETs MAY include this media type in the "typ" header parameter of the JWT representing the SET to explicitly declare that the JWT is a SET. This MUST be included if the SET could be used in an application context in which it could be confused with other kinds of JWTs.

Per the definition of "typ" in Section 4.1.9 of [RFC7515], it is RECOMMENDED that the "application/" prefix be omitted. Therefore, the "typ" value used SHOULD be "secevent+jwt".

2.4. Security Event Token Construction

This section describes how to construct a SET.

The following is an example JWT Claims Set for a hypothetical SCIM SET (which has been formatted for readability):

```
{
  "iss": "https://scim.example.com",
  "iat": 1458496404,
  "jti": "4d3559ec67504aaba65d40b0363faad8",
  "aud": [
   "https://scim.example.com/Feeds/98d52461fa5bbc879593b7754",
   "https://scim.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],

  "events": {
    "urn:ietf:params:scim:event:create": {
      "ref":
        "https://scim.example.com/Users/44f6142df96bd6ab61e7521d9",
      "attributes": ["id", "name", "userName", "password", "emails"]
    }
  }
}
```

Figure 5: Example Event Claims

The JSON Claims Set is encoded per [RFC7519].

In this example, the SCIM SET claims are encoded in an unsecured JWT. The JOSE Header for this example is:

```
{"typ":"secevent+jwt","alg":"none"}
```

Base64url encoding of the octets of the UTF-8 representation of the JOSE Header yields:

```
eyJ0eXAiOiJzZWNldmVudCtqd3QiLCJhbGciOiJub25lIn0
```

The above example JWT Claims Set is encoded as follows:

```
eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFkOCIsImlhdCI6MTQ1
ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiYXVkIjpbImh0
dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3
NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIxZDA4
NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50
OmNyZWF0ZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRm
NjE0MmRmOTZiZDZhYjYxZTc1MjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuYW1lIiwi
dXNlck5hbWUiLCJwYXNzd29yZCIsImVtYWlscyJdfX19
```

The encoded JWS signature is the empty string. Concatenating the parts yields this complete SET:

eyJ0eXAiOiJzZWNldmVudCtqd3QiLCJhbGciOiJub25lIn0.
eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFkOCIsImlhdCI6MTQ1
ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiYXVkIjpbImh0
dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3
NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIxZDA4
NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50
OmNyZWF0ZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRm
NjE0MmRmOTZiZDZhYjYxZTc1MjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuYW1lIiwi
dXNlck5hbWUiLCJwYXNzd29yZCIsImVtYWlscyJdfX19.

Figure 6: Example Unsecured Security Event Token

For the purpose of having a simpler example in Figure 6, an unsecured token is shown. When SETs are not signed or encrypted, other mechanisms such as TLS MUST be employed to provide integrity, confidentiality, and issuer authenticity, as needed by the application.

When validation (i.e., auditing), or additional transmission security is required, JWS signing and/or JWE encryption MAY be used. To create and or validate a signed and/or encrypted SET, follow the instructions in Section 7 of [RFC7519].

3. Requirements for SET Profiles

Profiling specifications for SETs define the syntax and semantics of SETs conforming to that SET profile and rules for validating those SETs. The syntax defined by profiling specifications includes what claims and event payload values are used by SETs utilizing the profile.

Defining the semantics of the SET contents for SETs utilizing the profile is equally important. Possibly most important is defining the procedures used to validate the SET issuer and to obtain the keys controlled by the issuer that were used for cryptographic operations used in the JWT representing the SET. For instance, some profiles may define an algorithm for retrieving the SET issuer’s keys that uses the "iss" claim value as its input. Likewise, if the profile allows (or requires) that the JWT be unsecured, the means by which the integrity of the JWT is ensured MUST be specified.

Profiling specifications MUST define how the event subject is identified in the SET, as well as how to differentiate between the event subject’s issuer and the SET issuer, if applicable. It is NOT RECOMMENDED for profiling specifications to use the "sub" claim in

cases in which the subject is not globally unique and has a different issuer from the SET itself.

Among the syntax and semantics of SETs that a profiling specification may define is whether the value of the "events" claim may contain multiple members, and what processing instructions are employed in the single- and multiple-valued cases for SETs conforming to that profile. Many valid choices are possible. For instance, some profiles might allow multiple event identifiers to be present and specify that any that are not understood by recipients be ignored, thus enabling extensibility. Other profiles might allow multiple event identifiers to be present but require that all be understood if the SET is to be accepted. Some profiles might require that only a single value be present. All such choices are within the scope of profiling specifications to define.

Profiling specifications MUST clearly specify the steps that a recipient of a SET utilizing that profile MUST perform to validate that the SET is both syntactically and semantically valid.

4. Security Considerations

4.1. Confidentiality and Integrity

SETs may contain sensitive information. Therefore, methods for distribution of events SHOULD require the use of a transport-layer security mechanism when distributing events. Parties MUST support TLS 1.2 [RFC5246] or a higher version and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the client MUST perform a TLS/SSL server certificate check, per [RFC6125]. Implementation security considerations for TLS can be found in "Recommendations for Secure Use of TLS and DTLS" [RFC7525].

Security events distributed through third parties or that carry personally identifiable information SHOULD be encrypted using JWE [RFC7516] or secured for confidentiality by other means.

Unless integrity of the JWT is ensured by other means, it MUST be signed using JWS [RFC7515] so that the SET can be authenticated and validated by the SET recipient.

4.2. Delivery

This specification does not define a delivery mechanism for SETs. In addition to confidentiality and integrity (discussed above), implementers and profiling specifications MUST consider the consequences of delivery mechanisms that are not secure and/or not

assured. For example, while a SET may be end-to-end secured using JWE encrypted SETs, without TLS, there is no assurance that the correct endpoint received the SET and that it could be successfully processed.

4.3. Sequencing

This specification defines no means of ordering multiple SETs in a sequence. Depending on the type and nature of the events represented by SETs, order may or may not matter. For example, in provisioning, event order is critical -- an object cannot be modified before it is created. In other SET types, such as a token revocation, the order of SETs for revoked tokens does not matter. If, however, the event conveys a logged in or logged out status for a user subject, then order becomes important.

Profiling specifications and implementers SHOULD take caution when using timestamps such as "iat" to define order. Distributed systems will have some amount of clock skew. Thus, time by itself will not guarantee order.

Specifications profiling SET SHOULD define a mechanism for detecting order or sequence of events when the order matters. For example, the "txn" claim could contain an ordered value (e.g., a counter) that the issuer includes.

4.4. Timing Issues

When SETs are delivered asynchronously and/or out-of-band with respect to the original action that incurred the security event, it is important to consider that a SET might be delivered to a SET recipient in advance of or behind the process that caused the event. For example, a user having been required to log out and then log back in again, may cause a logout SET to be issued that may arrive at the same time as the user agent accesses a web site having just logged in. If timing is not handled properly, the effect would be to erroneously treat the new user session as logged out. Profiling specifications SHOULD be careful to anticipate timing and subject selection information. For example, it might be more appropriate to cancel a "session" rather than a "user". Alternatively, the specification could use timestamps that allow new sessions to be started immediately after a stated logout event time.

4.5. Distinguishing SETs from ID Tokens

Because [RFC7519] states that "all claims that are not understood by implementations MUST be ignored", there is a consideration that a SET might be confused with ID Token [OpenID.Core] if a SET is mistakenly

or maliciously used in a context requiring an ID Token. If a SET could otherwise be interpreted as a valid ID Token (because it includes the required claims for an ID Token and valid issuer and audience claim values for an ID Token) then that SET profile MUST require that the "exp" claim not be present in the SET. Because "exp" is a required claim in ID Tokens, valid ID Token implementations will reject such a SET if presented as if it were an ID Token.

Excluding "exp" from SETs that could otherwise be confused with ID Tokens is actually defense in depth. In any OpenID Connect contexts in which an attacker could attempt to substitute a SET for an ID Token, the SET would actually already be rejected as an ID Token because it would not contain the correct "nonce" claim value for the ID Token to be accepted in contexts for which substitution is possible.

Note that the use of explicit typing, as described in Section 2.3, will not achieve disambiguation between ID Tokens and SETs, as the ID Token validation rules do not use the "typ" header parameter value.

4.6. Distinguishing SETs from Access Tokens

OAuth 2.0 [RFC6749] defines access tokens as being opaque. Nonetheless, some implementations implement access tokens as JWTs. Because the structure of these JWTs is implementation-specific, ensuring that a SET cannot be confused with such an access token is therefore likewise, in general, implementation specific. Nonetheless, it is recommended that SET profiles employ the following strategies to prevent possible substitutions of SETs for access tokens in contexts in which that might be possible:

- Prohibit use of the "exp" claim, as is done to prevent ID Token confusion.
- Where possible, use a separate "aud" claim value to distinguish between the SET recipient and the protected resource that is the audience of an access token.
- Modify access token validation systems to check for the presence of the "events" claim as a means to detect security event tokens. This is particularly useful if the same endpoint may receive both types of tokens.
- Employ explicit typing, as described in Section 2.3, and modify access token validation systems to use the "typ" header parameter value.

4.7. Distinguishing SETs from other kinds of JWTs

JWTs are now being used in application areas beyond the identity applications in which they first appeared. For instance, the Session Initiation Protocol (SIP) Via Header Field [RFC8055] and Personal Assertion Token (PASSporT) [I-D.ietf-stir-passport] specifications both define JWT profiles that use mostly or completely different sets of claims than are used by ID Tokens. If it would otherwise be possible for an attacker to substitute a SET for one of these (or other) kinds of JWTs, then the SET profile must be defined in such a way that any substituted SET will result in its rejection when validated as the intended kind of JWT.

The most direct way to prevent confusion is to employ explicit typing, as described in Section 2.3, and modify applicable token validation systems to use the "typ" header parameter value. This approach can be employed for new systems but may not be applicable to existing systems.

Another way to ensure that a SET is not confused with another kind of JWT is to have the JWT validation logic reject JWTs containing an "events" claim unless the JWT is intended to be a SET. This approach can be employed for new systems but may not be applicable to existing systems.

For many use cases, the simplest way to prevent substitution is requiring that the SET not include claims that are required for the kind of JWT that might be the target of an attack. For example, for [RFC8055], the "sip_callid" claim could be omitted and for [I-D.ietf-stir-passport], the "orig" claim could be omitted.

In many contexts, simple measures such as these will accomplish the task, should confusion otherwise even be possible. Note that this topic is being explored in a more general fashion in JSON Web Token Best Current Practices [I-D.ietf-oauth-jwt-bcp]. The proposed best practices in that draft may also be applicable for particular SET profiles and use cases.

5. Privacy Considerations

If a SET needs to be retained for audit purposes, the signature can be used to provide verification of its authenticity.

SET issuers SHOULD attempt to specialize SETs so that their content is targeted to the specific business and protocol needs of the intended SET recipients.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, SET issuers and recipients MUST have the appropriate legal agreements and user consent and/or terms of service in place.

The propagation of subject identifiers can be perceived as personally identifiable information. Where possible, SET issuers and recipients SHOULD devise approaches that prevent propagation -- for example, the passing of a hash value that requires the SET recipient to know the subject.

In some cases, it may be possible for a SET recipient to correlate different events and thereby gain information about a subject that the SET issuer did not intend to share. For example, a SET recipient might be able to use "iat" values or highly precise "toe" values to determine that two otherwise un-relatable events actually relate to the same real-world event. The union of information from both events could allow a SET recipient to de-anonymize data or recognize that unrelated identifiers relate to the same individual. SET issuers SHOULD take steps to minimize the chance of event correlation, when such correlation would constitute a privacy violation. For instance, they could use approximate values for the "toe" claim or arbitrarily delay SET issuance, where such delay can be tolerated.

6. IANA Considerations

6.1. JSON Web Token Claims Registration

This specification registers the "events", "toe", and "txn" claims in the IANA "JSON Web Token Claims" registry [IANA.JWT.Claims] established by [RFC7519].

6.1.1. Registry Contents

- Claim Name: "events"
- Claim Description: Security Events
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this specification]]

- Claim Name: "toe"
- Claim Description: Time of Event
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this specification]]

- Claim Name: "txn"
- Claim Description: Transaction Identifier
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this specification]]

6.2. Media Type Registration

6.2.1. Registry Contents

This section registers the "application/secevent+jwt" media type [RFC2046] in the "Media Types" registry [IANA.MediaTypes] in the manner described in [RFC6838], which can be used to indicate that the content is a SET.

- Type name: application
- Subtype name: secevent+jwt
- Required parameters: n/a
- Optional parameters: n/a
- Encoding considerations: 8bit; A SET is a JWT; JWT values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters.
- Security considerations: See the Security Considerations section of [[this specification]]
- Interoperability considerations: n/a
- Published specification: Section 2.3 of [[this specification]]
- Applications that use this media type: TBD
- Fragment identifier considerations: n/a
- Additional information:

  Magic number(s): n/a
  File extension(s): n/a
  Macintosh file type code(s): n/a

- Person & email address to contact for further information: Michael B. Jones, mbj@microsoft.com
- Intended usage: COMMON
- Restrictions on usage: none
- Author: Michael B. Jones, mbj@microsoft.com
- Change controller: IESG
- Provisional registration? No

7. References

7.1. Normative References

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

[IANA.MediaTypes]
IANA, "Media Types",
<http://www.iana.org/assignments/media-types>.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

[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>.

[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>.

[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>.

7.2. Informative References

[I-D.ietf-oauth-jwt-bcp]
Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", draft-ietf-oauth-jwt-bcp-00 (work in progress), July 2017.

[I-D.ietf-stir-passport]
Wendt, C. and J. Peterson, "Personal Assertion Token (PASSporT)", draft-ietf-stir-passport-11 (work in progress), February 2017.

[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>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

[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>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <https://www.rfc-editor.org/info/rfc7644>.

[RFC8055] Holmberg, C. and Y. Jiang, "Session Initiation Protocol (SIP) Via Header Field Parameter to Indicate Received Realm", RFC 8055, DOI 10.17487/RFC8055, January 2017, <https://www.rfc-editor.org/info/rfc8055>.

[RISC] OpenID Foundation, "OpenID Risk and Incident Sharing and Coordination (RISC) Working Group", <http://openid.net/wg/risc/>.

Appendix A. Acknowledgments

The editors would like to thank the members of the IETF SCIM working group, which began discussions of provisioning events starting with draft-hunt-scim-notify-00 in 2015.

The editors would like to thank the participants in the IETF id-event mailing list, the Security Events working group, and related working groups for their contributions to this specification.

Appendix B. Change Log

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

From the original draft-hunt-idevent-token:

Draft 01 - PH - Renamed eventUris to events

Draft 00 - PH - First Draft

Draft 01 - PH - Fixed some alignment issues with JWT. Remove event type attribute.

Draft 02 - PH - Renamed to Security Events, removed questions, clarified examples and intro text, and added security and privacy section.

Draft 03 - PH

General edit corrections from Sarah Squire

Changed "event" term to "SET"

Corrected author organization for William Denniss to Google

Changed definition of SET to be 2 parts, an envelope and 1 or more payloads.

Clarified that the intent is to express a single event with optional extensions only.

- mbj - Registered "events" claim, and proof-reading corrections.

Draft 04 - PH -

- Re-added the "sub" claim with clarifications that any SET type may use it.
- Added additional clarification on the use of envelope vs. payload attributes
- Added security consideration for event timing.
- Switched use of "attribute" to "claim" for consistency.
- Revised examples to put "sub" claim back in the top level.
- Added clarification that SETs typically do not use "exp".
- Added security consideration for distinguishing Access Tokens and SETs.

Draft 05 - PH - Fixed find/replace error that resulted in claim being spelled claimc

Draft 06 - PH -

- Corrected typos
- New txn claim
- New security considerations Sequencing and Timing Issues

Draft 07 -

- PH - Moved payload objects to be values of event URI attributes, per discussion.
- mbj - Applied terminology consistency and grammar cleanups.

Draft 08 - PH -

- Added clarification to status of examples
- Changed from primary vs. extension to state that multiple events may be expressed, some of which may or may not be considered extensions of others (which is for the subscriber or profiling specifications to determine).
- Other editorial changes suggested by Yaron

From draft-ietf-secevent-token:

Draft 00 - PH - First WG Draft based on draft-hunt-idevent-token

Draft 01 - PH - Changes as follows:

- Changed terminology away from pub-sub to transmitter/receiver based on WG feedback
- Cleaned up/removed some text about extensions (now only used as example)
- Clarify purpose of spec vs. future profiling specs that define actual events

Draft 02 - Changes are as follows:

- mbj - Added the Requirements for SET Profiles section.
- mbj - Expanded the Security Considerations section to describe how to prevent confusion of SETs with ID Tokens, access tokens, and other kinds of JWTs.
- mbj - Registered the "application/secevent+jwt" media type and defined how to use it for explicit typing of SETs.
- mbj - Clarified the misleading statement that used to say that a SET conveys a single security event.
- mbj - Added a note explicitly acknowledging that some SET profiles may choose to convey event subject information in the event payload.
- PH - Corrected encoded claim example on page 10.
- mbj - Applied grammar corrections.

Draft 03 - Changes are as follows:

- pjh - Corrected old "subscriber" to "Event Receiver". Added clarification in definition that Event Receiver is the same as JWT recipient.
- pjh - Added definition for "toe" (and IANA registration).
- pjh - Removed "nbf" claim.
- pjh - Figure 3, moved "sub" to the events payload next to "iss".
- pjh - Clarified the use of "nonce" in contexts where substitution is possible.

- mbj - Addressed WGLC comments by Nat Sakimura.

- mbj - Addressed WGLC comments by Annabelle Backman.

- mbj - Addressed WGLC comments by Marius Scurtescu.

Draft 04 - mbj - Changes were as follows:

- Clarified that all "events" values must represent aspects of the same state change that occurred to the subject -- not an aggregation of unrelated events about the subject.

- Removed ambiguities about the roles of multiple "events" values and the responsibilities of profiling specifications for defining how and when they are used.

- Corrected places where the term JWT was used when what was actually being discussed was the JWT Claims Set.

- Addressed terminology inconsistencies. In particular, standardized on using the term "issuer" to align with JWT terminology and the "iss" claim. Previously the term "transmitter" was sometimes used and "issuer" was sometimes used. Likewise, standardized on using the term "recipient" instead of "receiver" for the same reasons.

- Added a RISC event example, courtesy of Marius Scurtescu.

- Applied wording clarifications suggested by Annabelle Backman and Yaron Sheffer.

- Applied numerous grammar, syntax, and formatting corrections.

Draft 05 - mbj - Changes were as follows:

- Simplified the definitions of the "iat" and "toe" claims in ways suggested by Annabelle Backman.

- Added privacy considerations text suggested by Annabelle Backman.

- Updated the RISC event example, courtesy of Marius Scurtescu.

- Reordered the claim definitions to place the required claims first.

- Changed to using the RFC 8174 boilerplate instead of the RFC 2119 boilerplate.

Draft 06 - mbj - Changes were as follows:

- Changed "when the event was issued" to "when the SET was issued" in the "iat" description, as suggested by Annabelle Backman.

- Applied editorial improvements that improve the consistency of the specification that were suggested by Annabelle Backman, Marius Scurtescu, and Yaron Sheffer.

Draft 07 - PH - Text refinement to Section 3 proposed by Annabelle Backman post WGLC

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/

William Denniss
Google

Email: wdenniss@google.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com