Security Events
Internet-Draft
Intended status: Standards Track
Expires: 7 November 2026

A. Deshpande
A. Parecki
Okta

6 May 2026

Push-Based Delivery For Multiple Security Event Tokens (SET) Using HTTP
draft-deshpande-secevent-http-multi-set-push-02

Abstract

This specification defines how multiple Security Event Tokens (SETs) can be delivered to an intended recipient using HTTP POST over TLS. The SETs are 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.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://appsdesh.github.io/draft-deshpande-secevent-http-multi-set-push/draft-deshpande-secevent-http-multi-set-push.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-deshpande-secevent-http-multi-set-push/.

Discussion of this document takes place on the Security Events Working Group mailing list (mailto:id-event@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/id-event/. Subscribe at https://www.ietf.org/mailman/listinfo/id-event/.

Source for this draft and an issue tracker can be found at https://github.com/appsdesh/draft-deshpande-secevent-http-multi-set-push.

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/.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a mechanism by which a Transmitter of a Security Event Token (SET) [RFC8417] can deliver multiple SETs to an intended SET Recipient via HTTP POST [RFC7231] over TLS in a single POST request. [RFC8935] focuses on the delivery of the single SET to the Receiver. When sending a large number of SETs, sending them one by one is inefficient. This specification defines a way to send batches of SETs in a single POST request for more efficient transport.

Push-Based delivery for multiple SETs is intended to help in following scenarios:

* The Transmitter of the SET has multiple outstanding SETs to be communicated to the Receiver

* The Transmitter wants to reduce the number of outbound requests to the same Receiver to optimize performance, avoid being ratelimited when number of SETs to be communicated is high

* The Receiver wants to optimize processing multiple SETs

* The Receiver wants to acknowledge or provide error responses to previously received SETs, but wants to do so asynchronously, rather than within the response to the same HTTP POST in which it received the SET

This specification will handle all the use cases and scenarios for the [RFC8935] and make it more extensible to support multiple SETs per one outbound POST request.

Similar to [RFC8935] this specification makes 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 but is expected to be defined by profiles of this specification.

2. Conventions and Definitions

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.

3. Push endpoint to receive multiple SETs

Each Receiver that supports this specification MUST support a new push endpoint that receives multiple SETs in a single request. This endpoint MUST be capable of serving HTTP POST [RFC7231] requests. This endpoint MUST be TLS [RFC8446] enabled and MUST reject any communication not using TLS. The Transmitter obtains this endpoint from the Receiver is outside the scope of this specification.

4. SET Delivery Semantics

In this SET delivery using HTTP over TLS, a Transmitter delivers zero or more SETs in a JavaScript Object Notation (JSON) [RFC8259] document to the SET Receiver. The Receiver either acknowledges the successful receipt of the SETs or indicates failure in processing of one or more SETs in a JSON document to the Transmitter.

The Transmitter SHOULD periodically send a request with zero SETs to allow the Receiver to respond back with an ack or err for previously transmitted SETs that have not yet been acknowledged.

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 acknowledged (successful or failure) for over too long a period of time or if an excessive amount of storage is needed to retain them. If a Transmitter receives an acknowledgement or error for a SET it has no record of, the Transmitter MUST ignore that acknowledgement or error.

Upon receiving a SET, the SET Recipient reads the SET and validates it in the manner described in Section 2 of [RFC8935]. The SET Recipient MUST acknowledge receipt to the SET Transmitter, and SHOULD do so in a timely fashion (e.g., miliseconds). 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.

4.1. Acknowledgement for all SETs

A Receiver MUST ensure that it includes the jti value of each SET it receives, either in an ack or a setErrs value, to the Transmitter from which it received the SETs. A Transmitter SHOULD retry sending the same SET again if it was never responded to either in an ack value or in a setErrs value by a Receiver in a reasonable time period. A Transmitter MAY limit the number of times it retries

sending a SET. A Transmitter MAY publish the retry time period and maximum number of retries to its peers, but such publication is outside the scope of this specification.

4.2. Uniqueness of SETs

A Transmitter MUST NOT send two SETs with the same jti value if the SET has been either acknowledged through ack value or produced an error indicated by a setErrs value. If a Transmitter wishes to re-send an event after it has received a error response through a setErrs value, then it MUST generate a new SET that has a new (and unique) jti value.

4.3. Transmitting SETs

To transmit a SET to a SETs Recipient, the SET Transmitter makes an HTTP POST request to a TLS-enabled HTTP endpoint provided by the SET Recipient. The body of this request is of the content type "application/json" and the Accept header field MUST be "application/json".

A Transmitter may initiate communication with the Receiver in order to:

* Send SETs to the Receiver

* Receive acknowledgement of SETs in response

It MUST contain the following fields:

4.3.1. The sets Field

REQUIRED. A JSON object containing key-value pairs in which the key of a field is a string that contains the jti claim of the SET that is specified in the value of the field. This field MAY be an empty object to indicate that no SETs are being delivered by the initiator in this communication. The maximum number of SETs in a push MAY be set by the Transmitter for itself and SHOULD be communicated offline to the Receivers.

The following is a non-normative example of a request.

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

Figure 1: Example of SET Transmission

In the above example, the Transmitter is sending 2 SETs to the Receiver.

```
{
  "sets": {},
}
```

Figure 2: Example of empty SET transmission

In the above example, the Transmitter is sending zero SETs to the Receiver. This placeholder/empty request provides the Receiver to respond back with ack/err for previously transmitted SETs.

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 message descriptions.

4.4. Response Communication

A Receiver MUST repond to the communication by sending an HTTP response. The body of this response is of the content type "application/json". It contains the following fields:

ack REQUIRED. An array of strings, in which each string is the jti value of a previously received SET that is acknowledged in this object. This array MAY be empty to indicate that no previously received SETs are being acknowledged in this communication.

setErrs OPTIONAL. A JSON object containing key-value pairs in which the key of a field is a string that contains the jti value of a previously received SET that the sender of the communication object was unable to process. The value of the field is a JSON object that has the following fields:

err REQUIRED. The short reason why the specified SET failed to be processed. Error codes are described in Section 2.4 of [RFC8935].

description OPTIONAL. An explanation of why the SET failed to be processed

If the response contains a description, then 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].

4.4.1. Success Response

If the Receiver is successful in processing the request, it MUST return the HTTP status code 202 (Accepted). The response MUST have the content-type "application/json".

```
HTTP/1.1 202 Accepted
Content-type: application/json

{
  "ack": [
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ]
}
```

Figure 3: Example of SET Transmission response with ack

In the above example, the Receiver acknowledges one of the SETs it previously received. There are no errors reported by the Receiver.

```
HTTP/1.1 202 Accepted
Content-type: application/json

{
   "ack": [
    "f52901c499611ef94540242ac12000322",
    "0636e274399711ef9454-0242ac120002",
    "d563c72479a04ff0ba415657fa5e2cb11"
   ],
   "setErrs": {
    "4d3559ec67504aaba65d40b0363faad8" : {
      "err": "invalid_key",
      "description": "Failed validation"
    }
   }
}
```

Figure 4: Example of SET Transmission response, ack and errors

In the above example, the Receiver acknowledges three of the SETs it previously received. There are errors reported by the Receiver for acklowledging one SET.

4.4.2. 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, 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 REQUIRED. The short reason why the API failed to process the request. (Not specific to any SETs, but usually indicate service level failure or processing error)

description OPTIONAL. 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.

Note that failure responses in this specification are not specific to any failures related to any specific SET processing. SET specific errors should be communicated by a success response payload defined in the Section 4.4.1 Section.

Example error codes that can indicate API level failures MAY include but not limited to:

* invalid_request (request is malformed)

* authentication_failed (authentication token provided by the Transmitter is expired, revoked or invalid)

* access_denied (The Transmitter does not have adequate permissions to invoke this API).

* too_many_sets (Transmitter included too many SETs in a single request, this is an indication for the Transmitter to make a request with lower number of SETs or to comply with max SETs count that Receiver published outside of this spec)

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

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

_Figure 5: Example Error Response (authentication_failed)

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

4.4.2.1. Out of order delivery

A Response may contain jti values in its ack or setErrs that do not correspond to the SETs received in the same Request to which the Response is being sent. They MAY consist of values received in previous Requests.

4.4.3. Error Response

The Receiver MUST respond with an error response if it is unable to process the request. The error response MUST include the appropriate error code as described in Section 2.4 of [RFC8935].

5. Authentication and Authorization

The Transmitter MUST verify the identity of the Receiver by validating the TLS certificate presented by the Receiver during the TLS handshake, and verifying that it is the intended recipient of the request, before sending the SETs.

How the Transmitter and Receiver agree on authorization of the request is out of scope of this document.

This section describes server-side authentication of the Receiver by the Transmitter. Authentication of the Transmitter by the Receiver (e.g., via OAuth tokens, mutual TLS, or other mechanisms) is out of scope of this document and is expected to be defined by profiles of this specification.

6. Delivery Reliability

A Transmitter MUST attempt to deliver any SETs it has previously attempted to deliver to a Receiver until:

* It receives an acknowledgement through the ack value for that SET in a subsequent communication with the Receiver

* It receives a setErrs object for that SET in a subsequent communication with the Receiver

* It has attempted to deliver the SET a maximum number of times and has failed to communicate either due to communication errors or lack of inclusion in ack or setErrs in subsequent communications that were conducted for the maximum number of times. The maximum number of attempts MAY be set by the Transmitter for itself and SHOULD be communicated offline to the Receivers

Additionally consider Delivery Reliability aspects discussed in Section 4 of [RFC8935].

7. Security Considerations

The Security Considerations of [RFC8935], [RFC8446], and Section 17 of [RFC9110] apply to this specification.

7.1. Too many SETs in the request

This mechanism allows a Transmitter to send a large number of SETs in a single request. A malicious or misconfigured Transmitter could send an extremely large payload, attempting to exhaust memory or CPU resources on the Receiver during JSON parsing or SET validation.

Receivers MUST protect themselves against such attacks. It is RECOMMENDED that Receivers establish and document a reasonable upper limit on the number of SETs they will process in a single request. Transmitter MUST obey the maximum number of SETs to be communicated to the Receiver. This will avoid any potential truncations/loss of information at the Receiver.

If a Receiver receives a batch exceeding this limit, it SHOULD reject the entire request with a 413 Payload Too Large HTTP status code.

How the Receiver conveys this upper limit to Transmitters is outside the scope of this specification (see Section 4.3.1 for the sets field definition).

7.2. Authentication and Authorization

Transmitter MUST follow the procedures described in section Section 5 in order to securely authenticate and authorize Receiver

7.3. HTTP and TLS

Transmitter MUST use TLS [RFC8446] to communicate with Receiver and is subject to the security considerations of HTTP Section 17 of [RFC9110].

Failure to properly validate the Receiver's TLS certificate could allow a Transmitter to send SETs to an impersonating endpoint, resulting in the disclosure of sensitive security event information to an unauthorized party.

7.4. Event Delivery Latency

The primary purpose of security event tokens is the timely communication of security-sensitive information. While this specification enables batching for efficiency, Transmitters MUST NOT unduly delay the transmission of events in an attempt to create larger batches.

Delaying the transmission of a time-sensitive event, such as a credential compromise or session revocation, defeats the purpose of the protocol and provides an adversary with a larger window of opportunity to act.

It is RECOMMENDED that Transmitters implement a batching policy that sends a pending batch of SETs when either of the following conditions is met:

* The number of SETs in the batch reaches a configured size limit.

* A configured amount of time (e.g., 1-2 seconds) has elapsed since the oldest SET in the batch was generated.

This ensures a balance between network efficiency and the real-time nature of the communication.

7.5. Information Disclosure in Error Responses

The setErrs is designed for debugging and provides valuable feedback. However, if implemented incorrectly, it can become a souce of information leakage, disclosing internal details or enable enumeration type attacks.

It is RECOMMENDED that setErrs information be designed to be helpful without revealing sensitive information about internal architecture.

7.6. Event Ordering and Processing Guarantees

This specification is a transport efficiency mechanism and it does not address transactional aspects of the request. Every SET is an independent event in the request to the Receiver. The event ordering in the request does not imply any chronological dependence. For chronological dependence the Receiver should look at the time related event claims.

A Transmitter should not assume the ordered processing of the SETs by the Receiver sub-systems. This specification does not add any transactional requirements on the Receiver.

Additional security consideration in Section 5 of [RFC8935].

8. Privacy Considerations

Privacy Considerations from Section 6 of [RFC8935] apply.

9. IANA Considerations

This document has no IANA actions.

10. 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/rfc/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/rfc/rfc2277>.

[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/rfc/rfc7231>.

[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/rfc/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/rfc/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/rfc/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/rfc/rfc8446>.

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

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9728] Jones, M.B., Hunt, P., and A. Parecki, "OAuth 2.0 Protected Resource Metadata", RFC 9728, DOI 10.17487/RFC9728, April 2025, <https://www.rfc-editor.org/rfc/rfc9728>.

Acknowledgments

The authors would like to acknowledge the following individuals who contributed ideas, feedback, and wording that shaped and formed the final specification:

Atul Tulshibagwale, Yair Sarig, Yaron Sheffer.

Authors' Addresses

Apoorva Deshpande
Okta
Email: apoorva.deshpande@okta.com

Aaron Parecki
Okta
Email: aaron@parecki.com