Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 9, 2019

J. Bradley, Ed.
Yubico
T. Lodderstedt
YES.com AG
H. Zandbelt
ZmartZone

November 05, 2018

Encoding claims in the OAuth 2 state parameter using a JWT
draft-bradley-oauth-jwt-encoded-state-09

Abstract

This draft provides a method for a client to encode one or more elements encoding information about the session into the OAuth 2 "state" parameter.

Requirements Language

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 RFC 2119 [RFC2119].

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 May 9, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. The state JSON Web Token claims 3
3. Validating the state parameter 5
4. Creating a Request Forgery Protection (rfp) claim value. . . 5
4.1. Stateful Clients. 6
4.2. Stateless Clients. 6
4.3. Responses Initiated by the Issuer 6
5. IANA Considerations 6
6. Security Considerations 6
6.1. State Redirection 7
6.2. State Modification 7
6.3. State Confidentiality 7
6.4. Passing paramaters to the AS 7
7. Acknowledgements 7
8. Normative References 7
Appendix A. Document History 8
Authors' Addresses 8

1. Introduction

In the OAuth 2.0 Authorization protocol [RFC6749] , the Authorization server SHOULD perform an exact string comparison of the "redirect_uri" parameter with the "redirect_uri" parameter registered by by the client. This is essential for preventing token leakage to third parties in the OAuth implicit flow.

As a result of this clients can not safely add extra query parameters to the "redirect_uri" parameter that encode additional client state information.

The Client MUST use the "state" parameter to encode both Cross Site Request Forgery protection and any other state information it wishes to preserve for itself regarding the authorization request.

This draft proposes a mechanism whereby multiple state attributes can be encoded into a JSON Web Token JWT [RFC7519] for use as the value of the "state" parameter.

The JWT may be sent without integrity protection, with integrity protection JWS [RFC7515], or with both integrity and confidentiality protection JWE [RFC7516]. The client is free to choose the appropriate protection for its use-case as the "state" parameter is treated as opaque by the Authorization Server (AS).

2. The state JSON Web Token claims

The OAuth Authorization request "state" parameter consists of a JWT [RFC7519], optionally signed with JWS [RFC7515] or encrypted with JWE [RFC7516], whose payload contains claims as defined here.

rfp
: REQUIRED. string containing a verifiable identifier for the browser session, that cannot be guessed by a third party. The verification of this element by the client protects it from accepting authorization responses generated in response to forged requests generated by third parties.

kid
: RECOMMENDED if signed or encrypted. Identifier of the key used to sign this state token at the issuer. Identifier of the key used to encrypt this JWT state token at the issuer. This SHOULD be included in the JWE [RFC7516] header.

iat
: OPTIONAL. Timestamp of when this Authorization Request was issued.

exp
: OPTIONAL. The "exp" (expiration time) claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing. The processing of the "exp" claim requires that the current date/time MUST be before the expiration date/time listed in the "exp" claim. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. Its value MUST be a number containing an IntDate value. Use of this claim is OPTIONAL. This is RECOMMENDED if the JWT [RFC7519] state token is being produced by the AS.

iss
: OPTIONAL. string identifying the party that issued this state value.

aud
: OPTIONAL. string identifying the client that this state value is intended for.

target_link_uri
: OPTIONAL. URI containing the location the user agent is to be redirected to after authorization.

as
: OPTIONAL. string identifying the authorization server that this request was sent to.

jti
: RECOMMENDED. The "jti" (JWT ID) claim provides a unique identifier for the JWT. The identifier value MUST be assigned in a manner that ensures that there is a negligible probability that the same value will be accidentally assigned to a different data object. The "jti" claim can be used to prevent the JWT from being replayed. The "jti" value is a case-sensitive string. Use of this claim is OPTIONAL.

at_hash
: OPTIONAL. Access Token hash value. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the "access_token" value, where the hash algorithm used is the hash algorithm used in the "alg" parameter of the State Token's JWS [RFC7515] header. For instance, if the "alg" is "RS256", hash the "access_token" value with SHA-256, then take the left-most 128 bits and base64url encode them. The "at_hash" value is a case sensitive string. This is REQUIRED if the JWT [RFC7519] state token is being produced by the AS and issued with a "access_token" in the authorization response.

c_hash
: OPTIONAL. Code hash value. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the "code" value, where the hash algorithm used is the hash algorithm used in the "alg" header parameter of the State Token's JWS [RFC7515] header. For instance, if the "alg" is "HS512", hash the "code" value with SHA-512, then take the left-most 256 bits and base64url encode them. The "c_hash" value is a case sensitive string. This is REQUIRED if the JWT [RFC7519] state token is being produced by the AS and issued with a "code" in the authorization response.

The issuer may add additional claims to the token. The producer and the consumer of the JWT are the same or closely related entities so collision resistance of claim names should not be a concern.

The issuer SHOULD sign the JWT [RFC7519] with JWS [RFC7515] in such a way that it can verify the signature. The JWA [RFC7518] algorithm HS256 with a key of 256bits is recommended.

The issuer MAY sign the JWT [RFC7519] with JWS [RFC7515] using JWA [RFC7518] algorithm "none" if integrity protecting the contents of the "state" parameter is not required.

If the "state" parameter contains information the client doesn't want to disclose to the Authorization server or user, the issuer MAY encrypt the JWT [RFC7519] with JWE [RFC7516]. The JWA [RFC7518] algorithm ("alg") of "dir" and encryption algorithm ("enc") of "A128CBC-HS256" are recommended for symmetric encryption.

In the case of the "state" value being created by the Issuer the "iss" and "aud" claims MUST be included in the JWT. The jwt MUST also be signed with JWT [RFC7515]. If the State token is issued with a code "c_hash" MUST be included. If the State Token is issued with a Access Token "at_hash" MUST be included.

3. Validating the state parameter

Upon receiving a state parameter the client must validate its integrity. The client parses it as a JWT. It then verifies the signature of the JWT (if signed) using JWS [RFC7515]. The key used to sign the JWT [RFC7519] MAY be indicated by the kid field. The client MAY use other means to validate the JWT and determine its authenticity.

The client then reads the fields inside the JWT [RFC7519] and uses these to configure the user experience and security parameters of the authorization.

The "rfp" claim MUST be validated by the client by comparing it to the secret information that it used to create the "rfp" value.

The "as" claim if present MUST correspond to the URI endpoint registered as the "redirect_uri" for that AS.

4. Creating a Request Forgery Protection (rfp) claim value.

The client MUST create a value that cannot be guessed by a third party attacker and used to forge requests. There are many possible ways to create this value. For reference two common ways will be listed.

It is completely up to the purview of the particular client which generation methods, and which claims, they will accept.

4.1. Stateful Clients.

Many clients that are web servers maintain session state for browsers in a server side store.

These clients can generate a random value with sufficient entropy that an attacker cannot guess future values. This value can be stored in the server side store and used directly as the value of "rfp".

4.2. Stateless Clients.

Some clients that are web servers maintain session state for browsers using browser stored cookies or HTML5 local storage.

These clients can generate a hash value based on a HTTPS: bound session cookie or other browser side information that is not accessible to third parties. This hash value can be used as the value of "rfp".

While OAuth strongly recommends that clients use TLS to secure their endpoints, if a client is not using TLS it MUST produce the value of "rfp" by using a HMAC algorithm with a secret known only to itself over the browser stored information.

4.3. Responses Initiated by the Issuer

Some clients may be willing to rely on the Authorization server providing protection for Cross Site Request Forgery. In Cases where the Authorization server and the client have a pre-established relationship, and the client is willing to accept flows initiated by the Authorization server, the string "iss" may be used as the value of "rfp".

5. IANA Considerations

[maybe we register the "rfp" claim above?]

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

6.1. State Redirection

A client Authorization request might be redirected from the AS intended by the client, as part of an attack to confuse the client, and cause it to deliver a code or access tokens to a endpoint that the attacker can intercept them from. A Client that has multiple client_id issued by more than one AS SHOULD register a distinct "redirect_uri" value with each AS.

The "redirect_uri" that the Authorization response is received on MUST match the AS identified in the "as" claim.

If the AS allows pattern matching of query paramaters in the "redirect_uri" the identifier for the AS MUST be contained in the URI path component.

6.2. State Modification

Some information in the state JWT such as target_link_uri value for redirecting the user to the application might have some security impact is the user modifies them intentionally or unintentionally. To prevent tampering with the "state" value the client may integrity protect the contents of the JWT.

6.3. State Confidentiality

The client may have information that it wants to protect from disclosure to the Authorization server, in logs, to proxies, or to the user. In this case encrypting the JWT as a JWE is required to protect the confidentiality of the state information.

6.4. Passing paramaters to the AS

OAuth Signed Authorization request [I-D.ietf-oauth-signed-http-request] should be used to integrity protect paramaters sent by the client to the authorization server. The state paramater is intended for the consumption of the client, and to be opaque to the AS. Signed State should not be used to pass paramaters from the client to the AS.

7. Acknowledgements

8. Normative References

[I-D.ietf-oauth-signed-http-request]
Richer, J., Bradley, J., and H. Tschofenig, "A Method for Signing HTTP Requests for OAuth", draft-ietf-oauth-signed-http-request-03 (work in progress), August 2016.

[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>.

[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>.

[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>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[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>.

Appendix A. Document History

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

-05

o Added reference to OAuth Signed request

-04

o Updated references to JOSE/JWT

Authors' Addresses

John Bradley (editor)
Yubico

Email: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/

Dr.-Ing. Torsten Lodderstedt
YES.com AG

Email: torsten@lodderstedt.net

Hans Zandbelt
ZmartZone IAM

Email: hans.zandbelt@zmartzone.eu
URI: http://hanszandbelt.wordpress.com