OAuth Working Group N. Sakimura
Internet-Draft Nomura Research Institute
Intended status: Standards Track N. Matake
Expires: August 1, 2016 GREE, Inc.
SPR. Preibisch
CA Technologies
January 29, 2016

OAuth Response Metadata
draft-sakimura-oauth-meta-06

Abstract

This specification defines an extensible metadata that may be inserted into the OAuth 2.0 responses to assist the clients to process those responses. It is expressed either as a link header, or query parameters. It will allow the client to learn where the members in the response could be used, what is the characteristics of the payload is, how it should be processed, and so on. Since they are just additional response header/query parameters, any client that does not understand this extension should not break and work normally while supporting clients can utilize the metadata to take the advantage of the extension.

Table of Contents

1. Introduction

Although OAuth 2.0 [RFC6749] has been known for its REST friendliness, OAuth itself is not RESTful, as it heavily relies on out-of-band information to drive the interactions. This situation can be eased by hypertext-enabling the endpoint responses through the

introduction of data structure that represents such hypertext and other metadata.

Hyper-text enabling the OAuth responses has many advantages. For example,

- The authorization server can assign different token endpoints depending on the authorization request and the user authorization.

- It is possible to tell the client which resource endpoint it should use. This has a privacy advantage as well. The location of the resource by itself may be a sensitive information as its location may reveal information about the resource owner.

- It is possible to give a hint on the processing of the payload.

- It will be resitant to Mix-up attack.

- It will be resitant to Code Phishing Attack.

This specification defines methods to represent such metadata in the authorization and token responses.

1.1. Requirements Notation and 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 RFC 2119 [RFC2119].

1.2. terminology

This specification uses the terms "Access Token", "Authorization Code", "Authorization Endpoint", "Authorization Grant", "Authorization Server", "Client", "Client Authentication", "Protected Resource", "Refresh Token", and "Token Endpoint" defined by OAuth 2.0 [RFC6749].

2. Token Endpoint Response

Token Endpoints that implements this specification returns the following link relation (rel) and the corresponding URI value as defined in [RFC5988] in the Access Token Response defined in [RFC6749].

ruri Resource URI. The value of this parameter is the URI of the Resource Endpoint that the Access Token is supposed to be used at.
If this value is present, the cleint MUST NOT send the Access Token to any other URI.

turi Token Endpoint URI. The value of this parameter is the URI of the Token Endpoint that the Refresh Token can be sent to obtain a new Access Token. If this value is present, then the client MUST NOT send the refresh token to any other places.

Any other rels that are registered in Link Relation Type Registry defined in [RFC5988] registry can be used.

Following is an example of an HTTPS response.

```
   HTTP/1.1 200 OK
   Link: <https://example.com/userinfo>; rel="ruri",
    <https://example.com/payment-upon-trial-expiry>; rel="payments"
   Content-Type: application/JSON; charset=utf-8

   {
          "access_token":"aCeSsToKen"
   }
```

3. Authorization Response

While [RFC5988] defines a useful way of conveying link relations, it cannot be utilized for a redirect based communication such as the authorization response of OAuth 2.0. This section defines a way to return a limited set of those link relations as query parameters so that it can be conveyed over the redirection.

The authorization response of the implementation of this specification may return the following query parameter in the redirect URI.

turi Token Endpoint URI. The value of this parameter is the URI of the Token Endpoint that the Authorization Code can be sent to obtain the Access Token. If this parameter is specified, the client MUST check that the value of turi matches exactly with the pre-registered token endpoint URI of the Authorization Server that the session recoverd from the state variable points to. The cleint MUST NOT send the code to any other URIs than the value of this parameter.

ruri Resource URI. The value of this parameter is the URI of the Resource Endpoint that the Access Token can be used at. If this parameter is specified, the cleint MUST NOT send the Access Token to any other URIs than the value of this parameter.

As long as the link relation type string does not collide with the underlying protocol parameters, they can also be specified as a query parameter. The value MUST be encoded in application/x-www-form-urlencoded.

The following is an example of such resopnse. Line breaks are for display purposes only.

```
   HTTP/1.1 302 Found
   Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
       &turi=https%3A%2F%2Fexample.com%2Ftoken
       &state=xyz
```

4. IANA Considerations

4.1. Link Type Registration

Pursuant to [RFC5988], the following link type registrations [[will be]] registered by mail to link-relations@ietf.org.

- Relation Name: turi
- Description: An OAuth 2.0 Token Endpoint specified in section 3.2 of [RFC6749].
- Reference: This specification
- Relation Name: ruri
- Description: An OAuth 2.0 Resource Endpoint specified in section 3.2 of [RFC6750].
- Reference: This specification

5. Security Considerations

5.1. Authorization Response Query Parameter Tampering by a Bad User

The query response parameters may be tampered by the man-in-the-browser. It can also be tampered by a malicious user. In general, anything that comes via the browser/user-agent can be tainted and untrusted.

This specification mandates the turi check so that tamparing of turi by the malicious user will be detected. It does not mandate ruri check as the user can get the Access Token and send it to anywhere he wants anyways when it is returned to the browser.

However, other parameters are not protected. The Client MUST treat them tainted and implement its own check rules for each parameters.

To solve this "Tampering by bad user", either HMAC(concat(params)) need to be sent with them or have all of them inside the JWS.

6. Acknowledgements

Members of OAuth WG helped to form this specification. Notabely: Hannes tschofenig, John Bradley, Justin Richer, Kaoru Maeda, Masashi Kurabayashi, Michael B. Jones, Phil Hunt, William Dennis, James Manger, (add yourselves).

7. Document History

-06

- Removed duri description from token response as it is not needed.
- Made the processing instruction more precise.
- Added RFC5988 defined link relation type in the example.
- Swaped the order of the authorization response and token response.
  Now, token response gets explained first so that the reader will grasp the basic concept according to RFC5988 and regards the authorization response extension as a mapping of RFC5988 into query parameter form.

-05

- Factored out JSON Meta and now using query param and Web Linking.

-04

- Date refresh.

-03

- Date refresh.

-02

- Added Mike Kelly as an author.
- xref fix.
- Introduced "operations" as in draft-ietf-scim-api-00#section-3.5.
- Updated the informative reference to HAL.
- Added description to OAuth Token Endpoint hrefs.
- Added content-type to the example.
- Added Area and Working Group.

-01

- Some format changes, reference fix, and typo fixes.

- Changed 'items' to 'elements' to match the JSON terminology.

-00

- Initial Draft

8. References

8.1. Normative References

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

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

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

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

8.2. Informational References

[HAL] Kelly, M., "JSON Hypermedia API Language", February 2013.

[oauth-lrdd] Mills, W., "Link Type Registrations for OAuth 2", October 2012.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <http://www.rfc-editor.org/info/rfc4627>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

Authors' Addresses

Nat Sakimura
Nomura Research Institute

Email: sakimura@gmail.com

Nov Matake
GREE, Inc.

Email: nov@matake.jp
URI: http://matake.jp

Sascha Preibisch
CA Technologies

Email: Sascha.Preibisch@gmail.com