OAuth Working Group J. Richer, Ed.
Internet-Draft The MITRE Corporation
Intended status: Standards Track December 6, 2014
Expires: June 9, 2015

OAuth 2.0 Token Introspection
draft-ietf-oauth-introspection-03

Abstract

This specification defines a method for a protected resource to query an OAuth 2.0 authorization server to determine the active state of an OAuth 2.0 token and to determine meta-information about this token. OAuth 2.0 deployments can use this method to convey information about the authorization context of the token from the authorization server to the protected resource.

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 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 June 9, 2015.

Copyright Notice

Copyright (c) 2014 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 (http://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
  1.1. Terminology 3
2. Introspection Endpoint 3
  2.1. Introspection Request 4
  2.2. Introspection Response 5
  2.3. Error Response 6
3. IANA Considerations 6
4. Security Considerations 7
5. Privacy Considerations 8
6. Acknowledgements 8
7. References 8
  7.1. Normative References 8
  7.2. Informative References 9
Appendix A. Document History 9
Appendix B. Non-normative Examples 10
Appendix C. Use with Proof of Posession Tokens 12
Author's Address 12

1. Introduction

In OAuth 2.0, the contents of tokens are opaque to clients. This means that the client does not need to know anything about the content or structure of the token itself, if there is any. However, there is still a large amount of metadata that may be attached to a token, such as its current validity, approved scopes, and information about the context in which the token was issued. These pieces of information are often vital to protected resources making authorization decisions based on the tokens being presented. Since OAuth 2.0 [RFC6749] defines no direct relationship between the authorization server and the protected resource, only that they must have an agreement on the tokens themselves, there have been many different approaches to bridging this gap. These include using structured token formats such as JWT [JWT] or proprietary inter-service communication mechanisms (such as shared databases and protected enterprise service buses) that convey token information.

This specification defines an interoperable web API that allows authorized protected resources to query the authorization server to

determine the set of metadata for a given token that was presented to them by an OAuth 2.0 client. This metadata includes whether or not the token is currently active (or if it has expired or otherwise been revoked), what rights of access the token carries (usually conveyed through OAuth 2.0 scopes), and the authorization context in which the token was granted (including who authorized the token and which client it was issued to). Token introspection allows a protected resource to query this information regardless of whether or not it is carried in the token itself, allowing this method to be used along with or independently of structured token values. Additionally, a protected resource can use the mechanism described in this specification to introspect the token in a particular authorization decision context and ascertain the relevant metadata about the token in order to make this authorization decision appropriately.

1.1. Terminology

This section defines the terminology used by this specification. This section is a normative portion of this specification, imposing requirements upon implementations.

This specification uses the terms "access token", "authorization endpoint", "authorization grant", "authorization server", "client", "client identifier", "protected resource", "refresh token", "resource owner", "resource server", and "token endpoint" defined by OAuth 2.0 [RFC6749], and the terms "claim names" and "claim values" defined by JSON Web Token (JWT) [JWT].

2. Introspection Endpoint

The introspection endpoint is an OAuth 2.0 endpoint that takes a parameter representing an OAuth 2.0 token and returns a JSON [RFC7159] document representing the meta information surrounding the token, including whether this token is currently active. The definition of an active token is up to the authorization server, but this is commonly a token that has been issued by this authorization server, is not expired, has not been revoked, and is within the purview of the protected resource making the introspection call.

The introspection endpoint MUST be protected by TLS of at least version 1.2 RFC 5246 [RFC5246] and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the protected resource MUST perform a TLS/SSL server certificate check, per RFC 6125 [RFC6125]. Implementation security considerations can be found in Recommendations for Secure Use of TLS and DTLS [TLS.BCP].

2.1. Introspection Request

The protected resource calls the introspection endpoint using an HTTP POST [RFC7231] request with parameters sent as "application/x-www-form-urlencoded" data as defined in [W3C.REC-html5-20141028]. The authorization server MAY allow an HTTP GET [RFC7231] request with parameters passed in the query string as defined in [W3C.REC-html5-20141028]. The protected resource sends a parameter representing the token along with optional parameters representing additional context that is known by the protected resource to aid the authorization server in its response.

token REQUIRED. The string value of the token. For access tokens, this is the "access_token" value returned from the token endpoint defined in OAuth 2.0 [RFC6749] section 5.1. For refresh tokens, this is the "refresh_token" value returned from the token endpoint as defined in OAuth 2.0 [RFC6749] section 5.1. Other token types are outside the scope of this specification.

resource_id OPTIONAL. A service-specific string identifying the resource that the token is being used for. This value allows the protected resource to convey to the authorization server the context in which the token is being used at the protected resource, allowing the authorization server to tailor its response accordingly if desired.

token_type_hint OPTIONAL. A hint about the type of the token submitted for introspection. The protected resource re MAY pass this parameter in order to help the authorization server to optimize the token lookup. If the server is unable to locate the token using the given hint, it MUST extend its search across all of its supported token types. An authorization server MAY ignore this parameter, particularly if it is able to detect the token type automatically. Values for this field are defined in OAuth Token Revocation [RFC7009].

The endpoint MAY allow other parameters to provide further context to the query. For instance, an authorization service may need to know the IP address of the client accessing the protected resource in order to determine the appropriateness of the token being presented.

To prevent unauthorized token scanning attacks, the endpoint MUST also require some form of authorization to access this endpoint, such as client authentication as described in OAuth 2.0 [RFC6749] or a separate OAuth 2.0 access token such as the bearer token described in OAuth 2.0 Bearer Token Usage [RFC6750]. The methods of managing and validating these authentication credentials are out of scope of this specification.

2.2. Introspection Response

The server responds with a JSON object [RFC7159] in "application/json" format with the following top-level members. Several of these

active REQUIRED. Boolean indicator of whether or not the presented token is currently active. The authorization server determines whether and when a given token is in an active state.

scope OPTIONAL. A space-separated list of strings representing the scopes associated with this token, in the format described in section 3.3 of OAuth 2.0 [RFC6749].

client_id OPTIONAL. Client identifier for the OAuth 2.0 client that requested this token.

user_id OPTIONAL. Human-readable identifier for the user who authorized this token.

token_type OPTIONAL. Type of the token as defined in section 5.1 of OAuth 2.0 [RFC6749].

The response MAY include any claims defined as JWT [JWT] claim names and carry the same semantics and syntax, such as the following:

exp OPTIONAL. Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token will expire.

iat OPTIONAL. Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token was originally issued.

nbf OPTIONAL. Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token is not to be used before.

sub OPTIONAL. Subject of the token. Usually a machine-readable identifier of the resource owner who authorized this token

aud OPTIONAL. Service-specific string identifier or list of string identifiers representing the intended audience for this token.

iss OPTIONAL. String representing the issuer of this token.

jti OPTIONAL. String identifier for the token.

Specific implementations MAY extend this structure with their own service-specific pieces of information.

The response MAY be cached by the protected resource.

The authorization server MAY respond differently to different protected resources making the same request.

Note that in order to avoid disclosing too much of the authorization server's state to a third party, the authorization server SHOULD NOT include any additional information about an inactive token, including why the token is inactive.

2.3. Error Response

If the protected resource uses OAuth 2.0 client credentials to authenticate to the introspection endpoint and its credentials are invalid, the authorization server responds with an HTTP 401 (Unauthorized) as described in section 5.2 of OAuth 2.0 [RFC6749].

If the protected resource uses an OAuth 2.0 bearer token to authorize its call to the introspection endpoint and the token used for authorization does not contain sufficient privileges or is otherwise invalid for this request, the authorization server responds with an HTTP 401 code as described in section 3 of OAuth 2.0 Bearer Token Usage [RFC6750].

3. IANA Considerations

This specification requests IANA to register the following values into the IANA JSON Web Token Claims registry for JWT Claim Names.

- Claim Name: "active"
- Claim Description: Token active status
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this document]].

- Claim Name: "user_id"
- Claim Description: User identifier of the resource owner
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this document]].

- Claim Name: "client_id"
- Claim Description: Client identifier of the client
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this document]].

- Claim Name: "scope"
- Claim Description: Authorized scopes of the token
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this document]].

- Claim Name: "token_type"
- Claim Description: Type of the token
- Change Controller: IESG
- Specification Document(s): Section 2.2 of [[this document]].

4. Security Considerations

If left unprotected and un-throttled, the introspection endpoint could present a means for an attacker to poll a series of possible token values, fishing for a valid token. To prevent this, the authorization server MUST require authentication of protected resources that need to access the introspection endpoint and SHOULD require protected resources to be specifically authorized to call the introspection endpoint. The specifics of this authentication credentials are out of scope of this specification, but commonly these credentials could take the form of any valid client authentication mechanism used with the token endpoint, an OAuth 2.0 access token, or other HTTP authorization or authentication mechanism. A single piece of software acting as both a client and a protected resource MAY re-use the same credentials between the token endpoint and the introspection endpoint, though doing so potentially conflates the activities of the client and protected resource portions of the software and the authorization server MAY require separate credentials for each mode.

Since the introspection endpoint takes in OAuth 2.0 tokens as parameters, the server MUST support TLS 1.2 RFC 5246 [RFC5246] and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the client or protected resource MUST perform a TLS/SSL server certificate check, per RFC 6125 [RFC6125]. Implementation security considerations can be found in Recommendations for Secure Use of TLS and DTLS [TLS.BCP].

In order to prevent the values of access tokens being from leaking into server-side logs via query parameters, an authorization server offering token introspection MAY disallow HTTP GET and instead require an HTTP POST method only to the introspection endpoint.

In order to avoid disclosing internal server state, an introspection response for an inactive token SHOULD NOT contain any additional claims beyond the required "active" claim (with its value set to "false").

An authorization server offering token introspection MUST be able to understand the token values being presented to it during this call. The exact means by which this happens is an implementation detail and outside the scope of this specification. For unstructured tokens, this could take the form of a simple server-side database query

against a data store containing the context information for the token. For structured tokens, this could take the form of the server parsing the token, validating its signature or other protection mechanisms, and returning the information contained in the token back to the protected resource (allowing the protected resource to be unaware of the token's contents, much like the client).

Note that for tokens carrying encrypted information that is needed during the introspection process, the authorization server MUST be able to decrypt and validate the token in order to access this information. Also note that in cases where the authorization server stores no information about the token and has no means of accessing information about the token by parsing the token itself, it can not likely offer an introspection service.

5. Privacy Considerations

The introspection response may contain privacy-sensitive information such as user identifiers for resource owners. When this is the case, measures MUST be taken to prevent disclosure of this information to unintended parties. One way to limit disclosure is to require authorization to call the introspection endpoint and to limit calls to only registered and trusted protected resource servers. Another method is to transmit user identifiers as opaque service-specific strings, potentially returning different identifiers to each protected resource. Omitting privacy-sensitive information from an introspection response is the simplest way of minimizing privacy issues.

6. Acknowledgements

Thanks to the OAuth Working Group and the UMA Working Group for feedback.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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, March 2011.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

[RFC7009] Lodderstedt, T., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, August 2013.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[W3C.REC-html5-20141028]
Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E., O&#039;Connor, E., and S. Pfeiffer, "HTML5", World Wide Web Consortium Recommendation REC-html5-20141028, October 2014, <http://www.w3.org/TR/2014/REC-html5-20141028>.

7.2. Informative References

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token (work in progress), July 2014.

[TLS.BCP] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of TLS and DTLS", November 2014.

Appendix A. Document History

[[To be removed by the RFC Editor.]]

- 03

o Updated HTML and HTTP references.

o Call for registration of parameters in the JWT registry.

- 02

- Removed SAML pointer.

- Clarified what an "active" token could be.

- Explicitly declare introspection request as x-www-form-urlencoded format.

- Added extended example.

- Made protected resource authentication a MUST.

- 01

- Fixed casing and consistent term usage.

- Incorporated working group comments.

- Clarified that authorization servers need to be able to understand the token if they're to introspect it.

- Various editorial cleanups.

- 00

- Created initial IETF drafted based on draft-richer-oauth-introspection-06 with no normative changes.

Appendix B. Non-normative Examples

In this non-normative example, a protected resource receives a request from a client carrying an OAuth 2.0 bearer token as defined in OAuth 2.0 Bearer Token Usage [RFC6750]. In order to know how and whether to serve the request given this token, the protected resource then makes the following request to the introspection endpoint of the authorization server. The protected resource authenticates with its own credentials, here re-using the format of client identifier and client secret conveyed as HTTP Basic authentication as per OAuth 2.0 [RFC6749] Section 2.3.1.

Following is a non-normative example request:

```
POST /introspect HTTP/1.1
Host: authserver.example.com
Content-type: application/x-www-form-urlencoded
Accept: application/json
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3

token=X3241Affw.4233-99JXJ
```

In this non-normative example, the protected resource also sends a resource identifier and token type hint to the authorization server to aid the authorization server's response:

Following is a non-normative example request (with line wraps for display purposes only):

```
POST /introspect HTTP/1.1
Host: authserver.example.com
Content-type: application/x-www-form-urlencoded
Accept: application/json
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3

token=X3241Affw.4233-99JXJ
&resource_id=rsid-2348e.2381k3
&token_type_hint=access_token
```

The authorization server validates the protected resource's credentials and looks up the information in the token. If the token is currently active and the authenticated protected resource is authorized to know information about this token, the authorization server returns the following JSON document.

Following is a non-normative example active token response (with line wraps for display purposes only):

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

{
 "active": true,
 "client_id":"s6BhdRkqt3",
 "scope": "read write dolphin",
 "sub": "2309fj32kl",
 "user_id": "jdoe",
 "aud": "https://example.org/protected-resource/*",
 "iss": "https://authserver.example.com/"
}
```

If the token presented is not currently active for any reason (for instance, it has been revoked by the resource owner, it has expired, or the protected resource is not allowed to ask about this particular token) but the authorization presented during the request is otherwise valid, the authorization server returns the following JSON document.

Following is a non-normative example response to an inactive or invalid token (with line wraps for display purposes only):

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

{
 "active": false
}
```

Appendix C. Use with Proof of Posession Tokens

With bearer tokens such as those defined by OAuth 2.0 Bearer Token Usage [RFC6750], the protected resource will have in its possession the entire secret portion of the token for submission to the introspection service. However, for proof-of-possession style tokens, the protected resource will have only a token identifier used during the request, along with the cryptographic signature on the request. The protected resource would be able to submit the token identifier to the authorization server's token endpoint in order to obtain the necessary key information needed to validate the signature on the request. The details of this usage are outside the scope of this specification and should be defined in an extension to this specification.

Author's Address

Justin Richer (editor)
The MITRE Corporation

Email: jricher@mitre.org