OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

J. Richer, Ed.
The MITRE Corporation

July 4, 2014

OAuth Token Introspection
draft-richer-oauth-introspection-06

Abstract

This specification defines a method for a client or protected resource to query an OAuth authorization server to determine meta-information about an OAuth token.

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 January 5, 2015.

Copyright Notice

Table of Contents

1. Introduction

In OAuth, 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 extra information about the authentication 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 OAuth2 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.

This specification defines an Introspection Endpoint that allows the holder of a token to query the Authorization Server to discover the set of metadata for a token. A Protected Resource may use the mechanism described in this draft to query the Introspection Endpoint in a particular authorization decision context and ascertain the relevant metadata about the token in order to make this authorization decision appropriately.

2. Introspection Endpoint

The Introspection Endpoint is an OAuth 2 Endpoint that responds to HTTP POST requests (and optionally HTTP GET requests) from token holders, particularly including Resource Servers and Clients. The endpoint takes a single parameter representing the token (and

optionally further authentication) and returns a JSON document representing the meta information surrounding the token.

The endpoint MUST be protected by TLS or equivalent.

2.1. Introspection Request

token REQUIRED. The string value of the token.

resource_id OPTIONAL. A service-specific string identifying the resource that the client doing the introspection is asking about.

token_type_hint OPTIONAL. A hint about the type of the token submitted for introspection. Clients 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 accross 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 context to the query. For instance, an authorization service may need to know the IP address of the Client in order to determine the appropriateness of the token being presented.

The endpoint SHOULD also require some form of authentication to access this endpoint, such as the 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 [RFC4627] in "application/json" format with the following top-level members. Specific implementations MAY extend this structure with their own service-specific pieces of information.

active REQUIRED. Boolean indicator of whether or not the presented token is currently active.

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.

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 Client that requested this token.

sub OPTIONAL. Machine-readable identifier local to the AS of the Resource Owner who authorized this token.

user_id OPTIONAL. Human-readable identifier for the user 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.

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

The response MAY be cached according to HTTP caching headers.

2.3. Non-normative Example

For example, a Protected Resource recieves a request from a Client carrying an OAuth2 Bearer Token. In order to know how and whether to serve the request, the Protected Resource then makes the following request to the Introspection Endpoint of the Authorization Server. The Protected Resource is here authenticating with its own Client ID and Client Secret as per OAuth2 [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
```


The Authorization Server validates the client credentials and looks up the information in the token. If the token is currently active, it 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
Cache-Control: no-store

{
 "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 (but the

authentication presented during the request is valid), it 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
Cache-Control: no-store

{
 "active": false
}
```

If the client credentials are invalid or there is another error, the Authorization Server responds with an HTTP 400 (Bad Request) as described in OAuth 2.0 section 5.2 [RFC6749].

3. IANA Considerations

This document makes no request of IANA.

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. Therefore, the Authorization Server SHOULD issue special client credentials to any protected resources or clients that need to access the introspection endpoint. These credentials may be used directly at the endpoint, or they may be exchanged for an OAuth2 Access token scoped specifically for the Introspection Endpoint.

Since the introspection endpoint takes in OAuth 2 tokens as parameters, it MUST be protected by TLS or equivalent.

In order to prevent the access tokens being introspected from leaking

into server-side logs via query parameters, a server MAY require an HTTP POST method only to the endpoint.

5. Acknowledgements

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

6. Normative References

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

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[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.

Author's Address

Justin Richer (editor)
The MITRE Corporation

Email: jricher@mitre.org