OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 9, 2013

T. Lodderstedt, Ed.
Deutsche Telekom AG
S. Dronia

M. Scurtescu
Google

October 6, 2012

Token Revocation
draft-ietf-oauth-revocation-01

Abstract

This draft proposes an additional endpoint for OAuth authorization servers for revoking tokens.

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 April 9, 2013.

Copyright Notice

Table of Contents

1. Introduction

The OAuth 2.0 core specification [I-D.ietf-oauth-v2] defines several ways for a client to obtain refresh and access tokens. This specification supplements the core specification with a mechanism to revoke both types of tokens. A token is the external representation of an access grant issued by a resource owner to a particular client. A revocation request may discard the actual token as well as other tokens based on the same access grant and the access grant itself.

This mechanism facilitates the following use cases:

- The end-user triggers revocation from within the client that sends the appropriate revocation request to the autorization server. The request causes the removal of the client's access grant the particular token refers to. From the end-user's perspective, this looks like a "logout" or "reset" function. This use case makes it even more comfortable to the end-user to revoke his access grant immediately via the client.

- In contrast to revocation by a client, the authorization server (or a related entity) may offer its end-users a self-care portal to delete access grants given to clients independent of any token storing devices. Such a portal offers the possibility to an end-user to look at and revoke all access grants he once authorized. In cases the token storing device is not available, e.g. it is lost or stolen, revocation by a self-care portal is the only possibility to limit or avoid abuse.

In the end, security, usability, and ease of use are increased by token revocation.

By using an additional endpoint, the token revocation endpoint, clients can request the revocation of a particular token. Compliant implementation MUST support the revocation of refresh tokens, access token revocation MAY be supported.

2. Token Revocation

The client requests the revocation of a particular token by making an HTTP POST request to the token revocation endpoint. The location of the token revocation endpoint can be found in the authorization server's documentation. The token endpoint URI MAY include a query component.

Since requests to the token revocation endpoint result in the transmission of plain text credentials in the HTTP request, the

authorization server MUST require the use of a transport-layer security mechanism when sending requests to the token revocation endpoints. The authorization server MUST support TLS 1.0 ([RFC2246]), SHOULD support TLS 1.2 ([RFC5246]) and its future replacements, and MAY support additional transport-layer mechanisms meeting its security requirements.

The client constructs the request by including the following parameters using the "application/x-www-form-urlencoded" format in the HTTP request entity-body:

token REQUIRED. The token that the client wants to get revoked. Note: the authorization server is supposed to detect the token type automatically.

The client also includes its authentication credentials as described in Section 2.3. of [I-D.ietf-oauth-v2].

For example, a client may request the revocation of a refresh token with the following request (line breaks are for display purposes only):

```
POST /revoke HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

token=45ghiukldjahdnhzdauz&
```

The authorization server first validates the client credentials (in case of a confidential client) and verifies whether the client is authorized to revoke the particular token based on the client identity and its policy. For example, only the client the token has been issued for might be allowed to revoke it. It is also conceivable to allow a dedicated user self-care portal to revoke all kinds of tokens.

In the next step, the authorization server invalidates the token and the respective access grant. If the particular token is a refresh token and the authorization server supports the revocation of access tokens, then the authorization server SHOULD also invalidate all access tokens based on the same access grant.

Whether the revocation takes effect instantly or with some delay depends on the architecture of the particular deployment. The client MUST NOT make any assumptions about the timing and MUST NOT use the token again.

The authorization server indicates a successful processing of the request by a HTTP status code 200. Status code 401 indicates a failed client authentication, whereas a status code 403 is used if the client is not authorized to revoke the particular token. For all other error conditions, a status code 400 is used along with an error response as defined in section 5.2. of [I-D.ietf-oauth-v2]. The following error codes are defined for the token revocation endpoint:

unsupported_token_type The authorization server does not support the revocation of the presented token type. I.e. the client tried to revoke an access token on a server not supporting this feature.

invalid_token The presented token is invalid.

2.1. Cross-Origin Support

The revokation end-point SHOULD support CORS [W3C.WD-cors-20120403] if it is aimed at use in combination with user-agent-based applications. In addition, for interoperability with legacy user-agents, it MAY offer JSONP [jsonp] by allowing GET requests with an additional parameter:

callback The qualified name of a JavaScript function.

Example request:

```
https://example.com/revoke?token=45ghiukldjahdnhzdauz&
callback=package.myCallback
```

Successful response:

```
package.myCallback();
```

Error response:

```
package.myCallback({"error":"invalid_token"});
```

Clients should be aware that when relying on JSONP, a malicious revokation end-point may attempt to inject malicious code into the client.

3. Acknowledgements

We would like to thank Michiel de Jong, Doug Foiles, Paul Madsen, George Fletcher, Sebastian Ebling, Christian Stuebner, Brian Campbell, Igor Faynberg, Lukas Rosenstock, and Justin P. Richer for

their valuable feedback.

4. IANA Considerations

This draft includes no request to IANA.

5. Security Considerations

All relevant security considerations have been given in the functional specification.

6. References

6.1. Normative References

[I-D.ietf-oauth-v2]
Hardt, D., "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-31 (work in progress), August 2012.

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

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

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

6.2. Informative References

[W3C.WD-cors-20120403]
Kesteren, A., "Cross-Origin Resource Sharing", World Wide Web Consortium LastCall WD-cors-20120403, April 2012, <http://www.w3.org/TR/2012/WD-cors-20120403>.

[jsonp] Ippolito, B., "Remote JSON - JSONP", December 2005.

Authors' Addresses

Torsten Lodderstedt (editor)
Deutsche Telekom AG

Email: torsten@lodderstedt.net

Stefanie Dronia

Email: sdronia@gmx.de

Marius Scurtescu
Google

Email: mscurtescu@google.com