OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 20, 2013

T. Lodderstedt, Ed.
Deutsche Telekom AG
S. Dronia
M. Scurtescu
Google

May 19, 2013

OAuth 2.0 Token Revocation
draft-ietf-oauth-revocation-09

Abstract

This document proposes an additional endpoint for OAuth authorization servers, which allows clients to notify the authorization server that a previously obtained refresh or access token is no longer needed. This allows the authorization server to cleanup security credentials. A revocation request will invalidate the actual token and, if applicable, other tokens based on the same authorization grant.

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 November 20, 2013.

Copyright Notice

Copyright (c) 2013 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
2. Token Revocation 3
2.1. Revocation Request 3
2.2. Revocation Response 5
2.2.1. Error Response 5
2.3. Cross-Origin Support 5
3. Implementation Note 6
4. IANA Considerations 7
4.1. OAuth Extensions Error Registration 7
4.1.1. The "unsupported_token_type" Error Value 7
4.1.2. OAuth Token Type Hint Registry 7
4.1.2.1. Registration Template 8
4.1.2.2. Initial Registry Contents 8
5. Security Considerations 8
6. Acknowledgements 9
7. References 9
7.1. Normative References 9
7.2. Informative References 10
Authors' Addresses 10

1. Introduction

The OAuth 2.0 core specification [RFC6749] 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 a string representing an authorization grant issued by the resource owner to the client. A revocation request will invalidate the actual token and, if applicable, other tokens based on the same authorization grant and the authorization grant itself.

From an end-user's perspective, OAuth is often used to log into a certain site or application. This revocation mechanism allows a client to invalidate its tokens if the end-user logs out, changes identity, or uninstalls the respective application. Notifying the authorization server that the token is no longer needed allows the authorization server to clean up data associated with that token (e.g. session data) and the underlying authorization grant. This behavior prevents a situation where there is still a valid authorization grant for a particular client which the end user is not aware of. This way, token revocation prevents abuse of abandoned tokens and facilitates a better end-user experience since invalidated authorization grants will no longer turn up in a list of authorization grants the authorization server might present to the end-user.

2. Token Revocation

Implementations MUST support the revocation of refresh tokens and SHOULD support the revocation of access tokens (see Implementation Note).

The client requests the revocation of a particular token by making an HTTP POST request to the token revocation endpoint URL. This URL MAY include a query component. The means to obtain the location of the revocation endpoint is out of scope of this specification. For example, the client developer may consult the server's documentation or automatic discovery may be used. As this endpoint is handling security credentials, the endpoint location needs to be obtained from a trustworthy source.

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.

2.1. Revocation Request

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.

token_type_hint OPTIONAL. A hint about the type of the token submitted for revocation. 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. This specification defines two such values:

* access_token: An Access Token as defined in [RFC6749] section 1.4

* refresh_token: A Refresh Token as defined in [RFC6749] section 1.5

Specific implementations, profiles, and extensions of this specification MAY define other values for this parameter parameter using the registry defined in Section 4.1.2.

The client also includes its authentication credentials as described in Section 2.3. of [RFC6749].

For example, a client may request the revocation of a refresh token with the following request:

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

token=45ghiukldjahdnhzdauz&token_type_hint=refresh_token
```

The authorization server first validates the client credentials (in case of a confidential client) and then verifies whether the token was issued to the client making the revocation request. If this validation fails, the request is refused and the client is informed of the error by the authorization server as described below.

In the next step, the authorization server invalidates the token. The client MUST assume the revocation is immediate upon the receipt of an HTTP 200 response from the server. The client MUST NOT use the token again after the revocation.

Depending on the authorization server's revocation policy, the revocation of a particular token may cause the revocation of related tokens and the underlying authorization 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 authorization grant (see Implementation Note). If the token passed to the request is an access token, the server MAY decide to revoke the respective refresh token as well.

Note: A client compliant with [RFC6749] MUST be prepared to handle unexpected token invalidation at any time. Independent of the revocation mechanism specified in this document, resource owners may decide to revoke authorization grants or the authorization server may invalidate tokens in order to mitigate security threats. Thus having different server policies with respect to cascading the revocation of tokens should not pose interoperability problems.

2.2. Revocation Response

The authorization server responds with HTTP status code 200 if the token has been revoked sucessfully or if the client submitted an invalid token. The content of the response body does not matter as all information is conveyed in the response code.

An invalid token type hint value is ignored by the authorization server and does not influence the revocation response.

2.2.1. Error Response

The error presentation conforms to the defintion in section 5.2 of [RFC6749]. The following additional error code is 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.

If the server responds with HTTP status code 503, the client must assume the token still exists and may retry after a reasonable delay. The server may include a "Retry-After" header in the response to indicate how long the service is expected to be unavailable to the requesting client.

2.3. Cross-Origin Support

The revocation end-point MAY 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 also offer JSONP [jsonp] by allowing GET requests with an additional parameter:

callback OPTIONAL. The qualified name of a JavaScript function.

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

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

Successful response:

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

Error response:

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

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

3. Implementation Note

OAuth 2.0 allows deployment flexibility with respect to the style of access tokens. The access tokens may be self-contained so that an resource server needs no further interaction with an authorization server issuing these tokens to perform an authorization decision of the client requesting access to a protected resource. A system design may, however, instead use access tokens that are handles referring to authorization data stored at the authorization server. This consequently requires a resource server to issue a request to the respective authorization server to retrieve the content of the access token every time a client presents an access token.

While these are not the only options they illustrate the implications for revocation. In the latter case the authorization server is able to revoke an access token previously issued to a client when the resource server relays a received access token. In the former case some (currently non-standardized) backend interaction between the authorization server and the resource server may be used when immediate access token revocation is desired. Another design

alternative is to issue short-lived access tokens, which can be refreshed at any time using the corresponding refresh tokens. This allows the authorization server to impose a limit on the time revoked access tokens are in use.

Which approach of token revocation is chosen will depend on the overall system design and on the application service provider's risk analysis. The cost of revocation in terms of required state and communication overhead is ultimately the result of the desired security properties.

4. IANA Considerations

4.1. OAuth Extensions Error Registration

This specification registers the following error values in the OAuth Extensions Error registry defined in [RFC6749].

4.1.1. The "unsupported_token_type" Error Value

Error name unsupported_token_type

Error usage location revocation endpoint error response

Related protocol extension Token Revocation Endpoint

Change controller IETF

Specification document(s) [this document]

4.1.2. OAuth Token Type Hint Registry

This specification establishes the OAuth Token Type Hint registry. Possible values of the parameter "token_type_hint" (see Section 2.1) are registered with a Specification Required ([RFC5226]) after a two-week review period on the TBD@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Expert(s) may approve registration once they are satisfied that such a specification will be published. Registration requests must be sent to the TBD@ietf.org mailing list for review and comment, with an appropriate subject (e.g., "Request for parameter: example"). Within the review period, the Designated Expert(s) will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. IANA must only accept registry updates from the Designated Expert(s) and should direct all requests for registration to the review mailing list.

4.1.2.1. Registration Template

Hint Value: The additional value, which can be used to indicate a certain token type to the authorization server.

Change controller: For Standards Track RFCs, state "IETF". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification document(s): Reference to the document(s) that specify the type, preferably including a URI that can be used to retrieve a copy of the document(s). An indication of the relevant sections may also be included but is not required.

4.1.2.2. Initial Registry Contents

The OAuth Token Type Hint registry's initial contents are:

- Hint Value: access_token
- Change controller: IETF
- Specification document(s): [this document]
- Hint Value: refresh_token
- Change controller: IETF
- Specification document(s): [this document]

5. Security Considerations

If the authorization server does not support access token revocation, access tokens will not be immediately invalidated when the corresponding refresh token is revoked. Deployments must take this into account when conducting their security risk analysis.

Cleaning up tokens using revocation contributes to overall security and privacy since it reduces the likelihood for abuse of abandoned tokens. This specification in general does not intend to provide countermeasures against token theft and abuse. For a discussion of respective threats and countermeasures, consult the security considerations given in section 10 of the OAuth core specification [RFC6749] and the OAuth threat model document [RFC6819].

Malicious clients could attempt to use the new endpoint to launch denial of service attacks on the authorization server. Appropriate

countermeasures, which should be in place for the token endpoint as well, MUST be applied to the revocation endpoint (see [RFC6819], section 4.4.1.11). Specifically, invalid token type hints may misguide the authorization server and cause additional database lookups. Care MUST be taken to prevent malicious clients from exploiting this feature to launch denial of service attacks.

A malicious client may attempt to guess valid tokens on this endpoint by making revocation requests against potential token strings. According to this specification, a client's request must contain a valid client_id, in the case of a public client, or valid client credentials, in the case of a confidential client. The token being revoked must also belong to the requesting client. If an attacker is able to successfully guess a public client's client_id and one of their tokens, or a private client's credentials and one of their tokens, they could do much worse damage by using the token elsewhere than by revoking it. If they chose to revoke the token, the legitimate client will lose its authorization grant and will need to prompt the user again. No further damage is done and the guessed token is now worthless.

Since the revocation endpoint is handling security credentials, clients need to obtain its location from a trustworthy source only. Otherwise, an attacker could capture valid security tokens by utilizing a counterfeit revocation endpoint. Moreover in order to detect counterfeit revocation endpoints, clients MUST authenticate the revocation endpoint (certificate validation etc.).

6. Acknowledgements

We would like to thank Peter Mauritius, Amanda Anganes, Mark Wubben, Hannes Tschofenig, Michiel de Jong, Doug Foiles, Paul Madsen, George Fletcher, Sebastian Ebling, Christian Stuebner, Brian Campbell, Igor Faynberg, Lukas Rosenstock, and Justin Richer for their valuable 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.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

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

7.2. Informative References

[RFC6819] Lodderstedt, T., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", RFC 6819, January 2013.

[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, <http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp>.

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