Network Working Group	M. Jones, Ed.
Internet-Draft	Microsoft
Intended status: Standards Track	E. Hammer-Lahav
Expires: June 3, 2011	Yahoo!
	D. Hardt
	independent
	D. Recordon
	Facebook
	November 30, 2010

TOC

The OAuth 2.0 Protocol: Bearer Tokens
draft-ietf-oauth-v2-bearer-00

Abstract

This specification describes how to use bearer tokens when accessing OAuth 2.0 protected resources.

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 3, 2011.

Copyright Notice

Table of Contents

1. Introduction

TOC

OAuth enables clients to access protected resources by obtaining an access token (a string that denotes a specific scope, duration, and other attributes), rather than using the resource owner's credentials.

Tokens are issued to third-party clients by an authorization server with the approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server. This specification describes how to make protected resource requests by treating an OAuth access token as a bearer token.

This specification defines the use of OAuth over HTTP (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.) [RFC2616] (or HTTP over TLS as defined by [RFC2818] (Rescorla, E., “HTTP Over TLS,” May 2000.)). Other specifications may extend it for use with other transport protocols.

TOC

1.1. Notational 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 [RFC2119] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

This document uses the Augmented Backus-Naur Form (ABNF) notation of [I-D.ietf-httpbis-p1-messaging] (Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing," March 2010.). Additionally, the following rules are included from [RFC2617] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June 1999.): realm, auth-param; from [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," January 2005.): URI-Reference; and from [I-D.ietf-httpbis-p1-messaging] (Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing," March 2010.): OWS, RWS, and quoted-string.

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

1.2. Terminology

TOC

All terms are as defined in The OAuth 2.0 Protocol (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, "The OAuth 2.0 Protocol," 2010.) [OAuth2].

1.3. Overview

TOC

OAuth provides a method for clients to access a protected resource on behalf of a resource owner. Before a client can access a protected resource, it must first obtain authorization (access grant) from the resource owner, then exchange the access grant for an access token (representing the grant's scope, duration, and other attributes). The client accesses the protected resource by presenting the access token to the resource server.

The access token provides an abstraction layer, replacing different authorization constructs (e.g. username and password, assertion) for a single token understood by the resource server. This abstraction enables issuing access tokens valid for a short time period, as well

as removing the resource server's need to understand a wide range of authentication schemes.

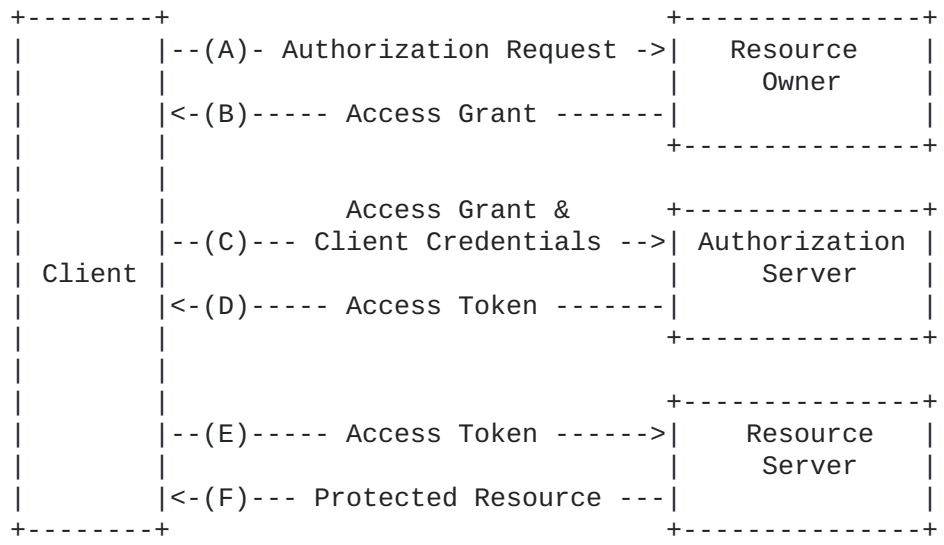

Figure 1: Abstract Protocol Flow

The abstract flow illustrated in Figure 1 (Abstract Protocol Flow) describes the overall OAuth 2.0 protocol architecture. The following steps are specified within this document:

> E) The client makes a protected resource request to the resource server by presenting the access token.
>
> F) The resource server validates the access token, and if valid, serves the request.

2. Authenticated Requests

TOC

Clients make authenticated token requests using the Authorization request header field. Resource servers MUST accept authenticated requests using the OAuth HTTP authentication scheme as described in Section 2.1 (The Authorization Request Header Field), and MAY support additional methods.

Alternatively, clients MAY attempt to include the access token using the HTTP request URI in the query component as described in Section 2.2 (URI Query Parameter), or in the HTTP body when using the application/x-www-form-urlencoded content type as described in

Section 2.3 (Form-Encoded Body Parameter). Resource server MAY support these alternative methods.

Clients SHOULD only use the request URI or body when the Authorization request header field is not available, and MUST NOT use more than one method in each request.

2.1. The Authorization Request Header Field

TOC

The Authorization request header field is used by clients to make authenticated token requests. The client uses the OAuth authentication scheme to include the access token in the request.

For example:

```
GET /resource HTTP/1.1
Host: server.example.com
Authorization: OAuth vF9dft4qmT
```

The Authorization header field uses the framework defined by [RFC2617] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, “HTTP Authentication: Basic and Digest Access Authentication,” June 1999.) as follows:

```
credentials    = "OAuth" RWS access-token [ RWS 1#auth-param ]
access-token   = 1*( quoted-char / <"> )

quoted-char    =   "!" / "#" / "$" / "%" / "&" / "'" / "("
                 / ")" / "*" / "+" / "-" / "." / "/" / DIGIT
                 / ":" / "<" / "=" / ">" / "?" / "@" / ALPHA
                 / "[" / "]" / "^" / "_" / "`" / "{" / "|"
                 / "}" / "~" / "\" / "," / ";"
```

> NOTE: [RFC5849] (Hammer-Lahav, E., “The OAuth 1.0 Protocol,” April 2010.) defines a different format for the OAuth authentication scheme. Resource servers can differentiate between the two protocol versions based on the presence of the oauth_signature_method which is REQUIRED in the previous version and is not supported by this specification.

2.2. URI Query Parameter

TOC

When including the access token in the HTTP request URI, the client adds the access token to the request URI query component as defined

by [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," January 2005.) using the oauth_token parameter.

For example, the client makes the following HTTP request using transport-layer security:

```
GET /resource?oauth_token=vF9dft4qmT HTTP/1.1
Host: server.example.com
```

The HTTP request URI query can include other request-specific parameters, in which case, the oauth_token parameters SHOULD be appended following the request-specific parameters, properly separated by an & character (ASCII code 38).

For example:

```
http://example.com/resource?x=y&oauth_token=vF9dft4qmT
```

> NOTE: The oauth_token parameter is used by the previous version of the OAuth protocol as described in [RFC5849] (Hammer-Lahav, E., "The OAuth 1.0 Protocol," April 2010.). Resource servers can differentiate between the two protocol versions based on the presence of the oauth_signature_method which is REQUIRED in the previous version and is not supported by this specification.

2.3. Form-Encoded Body Parameter

TOC

When including the access token in the HTTP request entity-body, the client adds the access token to the request body using the oauth_token parameter. The client can use this method only if the following REQUIRED conditions are met:

* The entity-body is single-part.

* The entity-body follows the encoding requirements of the application/x-www-form-urlencoded content-type as defined by [W3C.REC-html401-19991224] (Raggett, D., Hors, A., and I. Jacobs, "HTML 4.01 Specification," December 1999.).

* The HTTP request entity-header includes the Content-Type header field set to application/x-www-form-urlencoded.

* The HTTP request method is POST, PUT, or DELETE.

The entity-body can include other request-specific parameters, in which case, the oauth_token parameters SHOULD be appended following

the request-specific parameters, properly separated by an & character (ASCII code 38).

For example, the client makes the following HTTP request using transport-layer security:

```
POST /resource HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

oauth_token=vF9dft4qmT
```

> NOTE: The oauth_token parameter is used by the previous version of the OAuth protocol as described in [RFC5849] (Hammer-Lahav, E., "The OAuth 1.0 Protocol," April 2010.). Resource servers can differentiate between the two protocol versions based on the presence of the oauth_signature_method which is REQUIRED in the previous version and is not supported by this specification.

2.4. The WWW-Authenticate Response Header Field

TOC

If the protected resource request contains an invalid access token or is malformed, the resource server MUST include the HTTP WWW-Authenticate response header field. The WWW-Authenticate header field uses the framework defined by [RFC2617] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June 1999.) as follows:

```
challenge       = "OAuth" RWS token-challenge

token-challenge = realm
                  [ CS error ]
                  [ CS error-desc ]
                  [ CS error-uri ]
                  [ CS scope ]
                  [ CS 1#auth-param ]

error           = "error" "=" <"> token <">
error-desc      = "error_description" "=" quoted-string
error-uri       = "error_uri" = <"> URI-Reference <">
scope           = quoted-value /
                  <"> quoted-value *( 1*SP quoted-value ) <">
quoted-value    = 1*quoted-char

CS              = OWS "," OWS
```

For example:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: OAuth realm="Example Service",
error="invalid_token",
error_description="The access token expired"
```

The realm attribute is used to provide the protected resources partition as defined by [RFC2617] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June 1999.). [[add explanation]]

The error attribute is used to provide the client with the reason why the access request was declined. The parameter values are described in Section 2.5 (Error Codes).

The error_description attribute provides a human-readable text containing additional information, used to assist in the understanding and resolution of the error occurred.

The error_uri attribute provides a URI identifying a human-readable web page with information about the error, used to offer the end-user with additional information about the error. If the value is not an absolute URI, it is relative to the URI of the requested protected resource.

The scope attribute is a space-delimited list of scope values indicating the required scope of the access token for accessing the requested resource.

2.5. Error Codes

TOC

When a request fails, the resource server responds using the appropriate HTTP status code (typically, 400, 401, or 403), and includes one of the following error codes in the response:

> **invalid_request** The request is missing a required parameter, includes an unsupported parameter or parameter value, repeats the same parameter, uses more than one method for including an access token, or is otherwise malformed. The resource server SHOULD respond with the HTTP 400 (Bad Request) status code.
>
> **invalid_token** The access token provided is expired, revoked, malformed, or invalid for other reasons. The resource SHOULD respond with the HTTP 401 (Unauthorized) status code. The client MAY request a new access token and retry the protected resource request.
>
> **insufficient_scope** The request requires higher privileges than provided by the access token. The resource server SHOULD respond with the HTTP 403 (Forbidden) status code and MAY

> include the scope attribute with the scope necessary to access the protected resource.

[[Add mechanism for extending error codes]]

If the request lacks any authentication information (i.e. the client was unaware authentication is necessary or attempted using an unsupported authentication method), the resource server SHOULD not include an error code or other error information.

For example:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: OAuth realm="Example Service"
```

---

3. Security Considerations

TOC

Implementers and deployers must ensure that bearer tokens are not leaked to unintended parties, as they will be able to use them to gain access to protected resources. This is the primary security consideration when using bearer tokens with OAuth and underlies all the more specific statements that follow.

3.1. Validate SSL Certificate Chains

TOC

The client must validate the TLS certificate chain when making requests to protected resources. Failing to do so may enable DNS hijacking attacks to steal the token and gain unintended access.

3.2. Always use TLS (https)

TOC

Clients must always use TLS (https) when making requests with bearer tokens. Failing to do so exposes the token to numerous attacks that could give attackers unintended access.

3.3. Don't store bearer tokens in cookies

TOC

As cookies are generally sent in the clear, implementations must not store bearer tokens within them.

3.4. Issue short-lived bearer tokens

TOC

Using short-lived (one hour or less) bearer tokens can reduce the impact of one of them being leaked. The User-Agent flow should only issue short lived access tokens.

Appendix A. Acknowledgements

TOC

The following people contributed to preliminary versions of this document: Blaine Cook (BT), Brian Eaton (Google), Yaron Goland (Microsoft), Brent Goldman (Facebook), Raffi Krikorian (Twitter), Luke Shepard (Facebook), and Allen Tom (Yahoo!). The content and concepts within are a product of the OAuth community, WRAP community, and the OAuth Working Group.

The OAuth Working Group has dozens of very active contributors who proposed ideas and wording for this document, including: [[If your name is missing or you think someone should be added here, please send Mike Jones a note - don't be shy]]

Michael Adams, Andrew Arnott, Dirk Balfanz, Brian Campbell, Leah Culver, Bill de hÓra, Brian Ellin, Igor Faynberg, George Fletcher, Tim Freeman, Evan Gilbert, Justin Hart, John Kemp, Chasen Le Hara, Michael B. Jones, Torsten Lodderstedt, Eve Maler, James Manger, Laurence Miao, Chuck Mortimore, Justin Richer, Peter Saint-Andre, Nat Sakimura, Rob Sayre, Marius Scurtescu, Naitik Shah, Justin Smith, Jeremy Suriel, Christian Stübner, Paul Tarjan, and Franklin Tse.

Appendix B. Document History

TOC

[[to be removed by RFC editor before publication as an RFC]]

-00

*Initial draft based on preliminary version of OAuth 2.0 draft 11.

4. Normative References

TOC

[I-D.ietf-httpbis-p1-messaging]	Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing," draft-ietf-httpbis-p1-messaging-09 (work in progress), March 2010 (TXT).
[OAuth2]	Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, "The OAuth 2.0 Protocol," 2010.
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[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, June 1999 (TXT, PS, PDF, HTML, XML).
[RFC2617]	Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, June 1999 (TXT, HTML, XML).
[RFC2818]	Rescorla, E., "HTTP Over TLS," RFC 2818, May 2000 (TXT).
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," STD 66, RFC 3986, January 2005 (TXT, HTML, XML).
[RFC5849]	Hammer-Lahav, E., "The OAuth 1.0 Protocol," RFC 5849, April 2010 (TXT).
[W3C.REC-html401-19991224]	Raggett, D., Hors, A., and I. Jacobs, "HTML 4.01 Specification," World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 (HTML).

Authors' Addresses

TOC

	Michael B. Jones (editor)
	Microsoft
Email:	mbj@microsoft.com
URI:	http://self-issued.info/
	Eran Hammer-Lahav
	Yahoo!
Email:	eran@hueniverse.com
URI:	http://hueniverse.com
	Dick Hardt
	independent
Email:	dick.hardt@gmail.com
URI:	http://dickhardt.org/

	David Recordon
	Facebook
Email:	davidrecordon@facebook.com
URI:	http://www.davidrecordon.com/