Network Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	D. Hardt
Expires: November 22, 2011	independent
	D. Recordon
	Facebook
	May 21, 2011

TOC

The OAuth 2.0 Protocol: Bearer Tokens draft-ietf-oauth-v2-bearer-05

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 November 22, 2011.

Table of Contents

1. Introduction

TOC

OAuth enables clients to access protected resources by obtaining an access token, which is defined in **[I-D.ietf-oauth-v2]** as "a string representing an access authorization issued to the client", rather than using the resource owner's credentials.

Tokens are issued to 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 when the OAuth access token is a bearer token.

This specification defines the use of bearer tokens with OAuth over **HTTP** [RFC2616] using **TLS** [RFC5246]. Other specifications may extend it for use with other transport protocols.

1.1. Notational Conventions

TOC

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]**.

This document uses the Augmented Backus-Naur Form (ABNF) notation of **[I-D.ietf-httpbis-p1-messaging]**, which is based upon the Augmented Backus-Naur Form (ABNF) notation of **[RFC5234]**. Additionally, the following rules are included from **[RFC2617]**: auth-param and realm; from **[RFC3986]**: URI-Reference; and from **[I-D.ietf-httpbis-p1-messaging]**: RWS and quoted-string.

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

1.2. Terminology

TOC

Bearer Token
: A security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can. Using a bearer token does not require a bearer to prove possession of cryptographic key material (proof-of-possession).

All other terms are as defined in **[I-D.ietf-oauth-v2]**.

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 and 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.

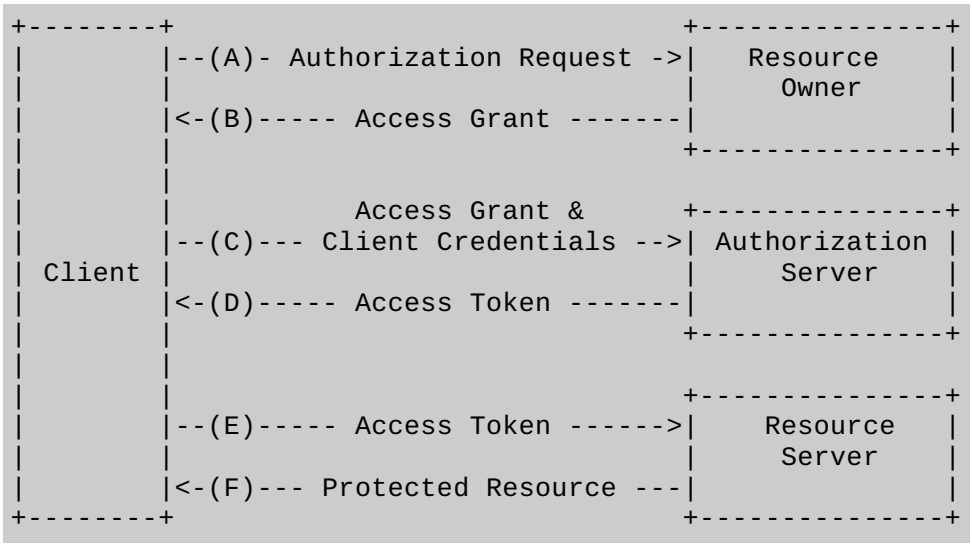

Figure 1: Abstract Protocol Flow

The abstract flow illustrated in **Figure 1** 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 SHOULD make authenticated requests with a bearer token using the `Authorization` request header field defined by **[RFC2617]**. Resource servers MUST accept authenticated requests using the `Bearer` HTTP authorization scheme as described in **Section 2.1**, and MAY support additional methods.

Alternatively, clients MAY transmit the access token in the HTTP body when using the `application/x-www-form-urlencoded` content type as described in **Section 2.2**; or clients MAY transmit the access token in the HTTP request URI in the query component as described in **Section 2.3**. Resource servers MAY support these alternative methods.

Clients SHOULD NOT use the request body or URI unless the `Authorization` request header field is not available, and MUST NOT use more than one method to transmit the token in each request. Because of the **Security Considerations** associated with the URI method, it SHOULD NOT be used unless no other method is feasible.

2.1. The Authorization Request Header Field

TOC

The `Authorization` request header field is used by clients to make authenticated requests with bearer tokens. The client uses the `Bearer` authentication scheme to transmit the access token in the request.

For example:

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

The `Authorization` header field uses the framework defined by **[RFC2617]** as follows:

```
credentials     = "Bearer" RWS access-token
access-token    = 1*( quoted-char / <"> )

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

TOC

2.2. Form-Encoded Body Parameter

When including the access token in the HTTP request entity-body, the client adds the access token to the request body using the `bearer_token` parameter. The client MUST NOT use this method unless the following conditions are met:

- The HTTP request 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]**.
- The HTTP request entity-header includes the `Content-Type` header field set to `application/x-www-form-urlencoded`.
- The HTTP request method is one for which a body is permitted to be present in the request. In particular, this means that the `GET` method MUST NOT be used.

The entity-body can include other request-specific parameters, in which case, the `bearer_token` parameter MUST be properly separated from the request-specific parameters 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

bearer_token=vF9dft4qmT
```

The `application/x-www-form-urlencoded` method SHOULD NOT be used except in application contexts where participating browsers do not have access to the `Authorization` request header field.

TOC

2.3. URI Query Parameter

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]** using the `bearer_token` parameter.

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

```
GET /resource?bearer_token=vF9dft4qmT HTTP/1.1
```

```
Host: server.example.com
```

The HTTP request URI query can include other request-specific parameters, in which case, the `bearer_token` parameter MUST be properly separated from the request-specific parameters by an `&` character (ASCII code 38).

For example:

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

Because of the **Security Considerations** associated with the URI method, it SHOULD NOT be used unless no other method is feasible.

TOC

2.4. The WWW-Authenticate Response Header Field

If the protected resource request does not include authentication credentials or contains an invalid access token, the resource server MUST include the HTTP `WWW-Authenticate` response header field; it MAY include it in response to other conditions as well. The `WWW-Authenticate` header field uses the framework defined by **[RFC2617]** as follows:

```
challenge       = "Bearer" [ RWS 1#param ]

param           = realm / scope /
                  error / error-desc / error-uri /
                  auth-param

scope           = "scope" "=" <"> scope-v *( SP scope-v ) <">
scope-v         = 1*quoted-char

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

error           = "error" "=" quoted-string
error-desc      = "error_description" "=" quoted-string
error-uri       = "error_uri" "=" <"> URI-reference <">
```

The `scope` attribute is a space-delimited list of scope values indicating the required scope of the access token for accessing the requested resource. The `scope` attribute MUST NOT appear more than once.

If the protected resource request included an access token and failed authentication, the resource server SHOULD include the `error` attribute to provide the client with the reason why the access request was declined. The parameter value is described in **Section 2.4.1**. In addition, the resource server MAY include the `error_description` attribute to provide a human-readable explanation, and the `error_uri` attribute with an absolute URI identifying a human-readable web page explaining the error. The `error`, `error_description`, and `error_uri` attribute MUST NOT appear more than once. [[add language and encoding information to error_description if the core specification does]]

For example, in response to a protected resource request without authentication:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example"
```

And in response to a protected resource request with an authentication attempt using an expired access token:

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

TOC

2.4.1. Error Codes

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 401 (Unauthorized) 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.

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: Bearer realm="example"
```

TOC

3. Security Considerations

This section describes the relevant security threats regarding token handling when using bearer tokens and describes how to mitigate these threats.

TOC

3.1. Security Threats

The following list presents several common threats against protocols utilizing some form of tokens. This list of threats is based on NIST Special Publication 800-63 **[NIST800-63]**. Since this document builds on the OAuth 2.0 specification, we exclude a discussion of threats that are described there or in related documents.

Token manufacture/modification:
: An attacker may generate a bogus token or modify the token contents (such as the authentication or attribute statements) of an existing token, causing the resource server to grant inappropriate access to the client. For example, an attacker may modify the token to extend the validity period; a malicious client may modify the assertion to gain access to information that they should not be able to view.

Token disclosure:
: Tokens may contain authentication and attribute statements that include sensitive information.

Token redirect:
: An attacker uses the token generated for consumption by resource server to obtain access to another resource server.

Token replay:
: An attacker attempts to use a token that has already been used with that resource server in the past.

TOC

3.2. Threat Mitigation

A large range of threats can be mitigated by protecting the contents of the token by using a digital signature or a Message Authentication Code (MAC). Alternatively, a bearer token can contain a reference to authorization information, rather than encoding the information directly. Such references MUST be infeasible for an attacker to guess; using a reference may require an extra interaction between a server and the token issuer to resolve the reference to the authorization information.

This document does not specify the encoding or the contents of the token; hence detailed recommendations for token integrity protection are outside the scope of this document. We assume that the token integrity protection is sufficient to prevent the token from being modified.

To deal with token redirect, it is important for the authorization server to include the identity of the intended recipients (the audience), typically a single resource server (or a list of resource servers), in the token. Restricting the use of the token to a specific scope is also recommended.

To provide protection against token disclosure, confidentiality protection is applied via **TLS** [RFC5246] with a ciphersuite that offers confidentiality protection. This requires that the communication interaction between the client and the authorization server, as well as the interaction between the client and the resource server, utilize confidentiality protection. Since TLS is mandatory to implement and to use with this specification, it is the preferred approach for preventing token disclosure via the communication channel. For those cases where the client is prevented from observing the contents of the token, token encryption has to be applied in addition to the usage of TLS protection.

To deal with token capture and replay, the following recommendations are made: First, the lifetime of the token has to be limited by putting a validity time field inside the protected part of the token. Note that using short-lived (one hour or less) tokens significantly reduces the impact of one of them being leaked. Second, confidentiality protection of the exchanges between the client and the authorization server and between the client and the resource server MUST be applied, for instance, through the use of **TLS** [RFC5246]. As a consequence, no eavesdropper along the communication path is able to observe the token exchange. Consequently, such an on-path adversary cannot replay the token. Furthermore, when presenting the token to a resource server, the client MUST verify the identity of that resource server, as per **[RFC2818]**. Note that the client MUST validate the TLS certificate chain when making these requests to protected resources. Presenting the token to an unauthenticated and unauthorized resource server or failing to validate the certificate chain will allow adversaries to steal the token and gain unauthorized access to protected resources.

TOC

3.3. Summary of Recommendations

Safeguard bearer tokens
: Client implementations 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 and underlies all the more specific recommendations that follow.

Validate SSL certificate chains
: 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.

Always use TLS (https)
: Clients MUST always use **TLS** [RFC5246] (https) when making requests with bearer tokens. Failing to do so exposes the token to numerous attacks that could give attackers unintended access.

Don't store bearer tokens in cookies
: Implementations MUST NOT store bearer tokens within cookies that can be sent in the clear (which is the default transmission mode for cookies).

Issue short-lived bearer tokens
: Using short-lived (one hour or less) bearer tokens can reduce the impact of one of them being leaked. In particular, only short-lived bearer tokens should be issued to clients that run within a web browser or other environments where information leakage may occur.

Issue scoped bearer tokens
: Issue bearer tokens that contain an audience restriction, scoping their use to the intended relying party or set of relying parties.

Don't pass bearer tokens in page URLs
: Browsers, web servers, and other software may not adequately secure URLs in the browser history, web server logs, and other data structures. If bearer tokens are passed in page URLs (typically as query string parameters), attackers might be able to steal them from the history data, logs, or other unsecured locations. Instead, pass bearer tokens in HTTP message headers or message bodies for which confidentiality measures are taken.

---

4. IANA Considerations

TOC

4.1. OAuth Access Token Type Registration

TOC

This specification registers the following access token type in the OAuth Access Token Type Registry.

4.1.1. The "Bearer" OAuth Access Token Type

TOC

Type name:
: Bearer

Additional Token Endpoint Response Parameters:
: (none)

HTTP Authentication Scheme(s):
: Bearer

Change controller:
: IETF

Specification document(s):
: [[this document]]

---

5. References

TOC

5.1. 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-14 (work in progress), April 2011 (**TXT**).
[I-D.ietf-oauth-v2]	Hammer-Lahav, E., Recordon, D., and D. Hardt, "**The OAuth 2.0 Authorization Protocol**," draft-ietf-oauth-v2-16 (work in progress), May 2011 (**TXT**, **PDF**).
[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).

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," STD 68, RFC 5234, January 2008 (TXT).

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

[W3C.REC-html401-19991224] Hors, A., Jacobs, I., and D. Raggett, "HTML 4.01 Specification," World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 (HTML).

5.2. Informative References

TOC

[I-D.ietf-httpbis-p7-auth] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 7: Authentication," draft-ietf-httpbis-p7-auth-13 (work in progress), March 2011 (TXT).

[NIST800-63] Burr, W., Dodson, D., Perlner, R., Polk, T., Gupta, S., and E. Nabbus, "NIST Special Publication 800-63-1, INFORMATION SECURITY," December 2008.

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, the 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: 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, Eran Hammer-Lahav, Chasen Le Hara, Michael B. Jones, Torsten Lodderstedt, Eve Maler, James Manger, Laurence Miao, Chuck Mortimore, Anthony Nadalin, 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 the RFC editor before publication as an RFC]]

-05

- Removed OAuth Errors Registry, per design team input.
- Changed HTTP status code for `invalid_request` error code from HTTP 400 (Bad Request) to HTTP 401 (Unauthorized) to match HTTP usage [[change pending working group consensus]].
- Added missing quotation marks in error-uri definition.
- Added note to add language and encoding information to error_description if the core specification does.
- Explicitly reference the Augmented Backus-Naur Form (ABNF) defined in **[RFC5234]**.
- Use auth-param instead of repeating its definition, which is (token "=" (token / quoted-string)).
- Clarify security considerations about including an audience restriction in the token and include a recommendation to issue scoped bearer tokens in the summary of recommendations.

-04

- Edits responding to working group last call feedback on -03. Specific edits

enumerated below.

- Added Bearer Token definition in Terminology section.
- Changed parameter name `oauth_token` to `bearer_token`.
- Added realm parameter to `WWW-Authenticate` response to comply with **[RFC2617]**.
- Removed "[RWS 1#auth-param]" from `credentials` definition since it did not comply with the ABNF in **[I-D.ietf-httpbis-p7-auth]**.
- Removed restriction that the `bearer_token` (formerly `oauth_token`) parameter be the last parameter in the entity-body and the HTTP request URI query.
- Do not require WWW-Authenticate Response in a reply to a malformed request, as an HTTP 400 Bad Request response without a WWW-Authenticate header is likely the right response in some cases of malformed requests.
- Removed OAuth Parameters registry extension.
- Numerous editorial improvements suggested by working group members.

-03

- Restored the WWW-Authenticate response header functionality deleted from the framework specification in draft 12 based upon the specification text from draft 11.
- Augmented the OAuth Parameters registry by adding two additional parameter usage locations: "resource request" and "resource response".
- Registered the "oauth_token" OAuth parameter with usage location "resource request".
- Registered the "error" OAuth parameter.
- Created the OAuth Error registry and registered errors.
- Changed the "OAuth2" OAuth access token type name to "Bearer".

-02

- Incorporated feedback received on draft 01. Most changes were to the security considerations section. No normative changes were made. Specific changes included:
- Changed terminology from "token reuse" to "token capture and replay".
- Removed sentence "Encrypting the token contents is another alternative" from the security considerations since it was redundant and potentially confusing.
- Corrected some references to "resource server" to be "authorization server" in the security considerations.
- Generalized security considerations language about obtaining consent of the resource owner.
- Broadened scope of security considerations description for recommendation "Don't pass bearer tokens in page URLs".
- Removed unused reference to OAuth 1.0.
- Updated reference to framework specification and updated David Recordon's e-mail address.
- Removed security considerations text on authenticating clients.
- Registered the "OAuth2" OAuth access token type and "oauth_token" parameter.

-01

- First public draft, which incorporates feedback received on -00 including enhanced Security Considerations content. This version is intended to accompany OAuth 2.0 draft 11.

-00

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

Authors' Addresses

TOC

Michael B. Jones
Microsoft
Email: mbj@microsoft.com
URI: http://self-issued.info/

Dick Hardt
independent

Email: dick.hardt@gmail.com
URI: http://dickhardt.org/

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