OAuth Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	D. Hardt
Expires: August 2, 2012	independent
	D. Recordon
	Facebook
	January 30, 2012

TOC

The OAuth 2.0 Authorization Protocol: Bearer Tokens
draft-ietf-oauth-v2-bearer-16

Abstract

This specification describes how to use bearer tokens in HTTP requests to access OAuth 2.0 protected resources. Any party in possession of a bearer token (a "bearer") can use it to get access to the associated resources (without demonstrating possession of a cryptographic key). To prevent misuse, bearer tokens need to be protected from disclosure in storage and in transport.

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 August 2, 2012.

Copyright Notice

Copyright (c) 2012 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**
   - 1.1. **Notational Conventions**
   - 1.2. **Terminology**
   - 1.3. **Overview**
2. **Authenticated Requests**
   - 2.1. **Authorization Request Header Field**
   - 2.2. **Form-Encoded Body Parameter**
   - 2.3. **URI Query Parameter**

3. The WWW-Authenticate Response Header Field
3.1. Error Codes
4. Security Considerations
4.1. Security Threats
4.2. Threat Mitigation
4.3. Summary of Recommendations
5. IANA Considerations
5.1. OAuth Access Token Type Registration
5.1.1. The "Bearer" OAuth Access Token Type
5.2. Authentication Scheme Registration
5.2.1. The "Bearer" Authentication Scheme
6. References
6.1. Normative References
6.2. Informative References
Appendix A. Acknowledgements
Appendix B. Document History
§ Authors' Addresses

---

1. Introduction

TOC

OAuth enables clients to access protected resources by obtaining an access token, which is defined in OAuth 2.0 Authorization **[I-D.ietf-oauth-v2]** as "a string representing an access authorization issued to the client", rather than using the resource owner's credentials directly.

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 over HTTP/1.1 **[I-D.ietf-httpbis-p1-messaging]** using TLS **[RFC5246]** to access protected resources. TLS is mandatory to implement and use with this specification; other specifications may extend this specification for use with other transport protocols. While designed for use with access tokens resulting from OAuth 2.0 Authorization **[I-D.ietf-oauth-v2]** flows to access OAuth protected resources, this specification actually defines a general HTTP authorization method that can be used with bearer tokens from any source to access any resources protected by those bearer tokens.

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 Key words for use in RFCs to Indicate Requirement Levels **[RFC2119]**.

This document uses the Augmented Backus-Naur Form (ABNF) notation of HTTP/1.1, Part 1 **[I-D.ietf-httpbis-p1-messaging]**, which is based upon the Augmented Backus-Naur Form (ABNF) **[RFC5234]** notation. Additionally, the following rules are included from HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]**: auth-param, auth-scheme, and b64token; and from Uniform Resource Identifier (URI) **[RFC3986]**: URI-Reference.

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 OAuth 2.0 Authorization **[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. In the general case, before a client can access a protected resource, it must first obtain an authorization grant from the resource owner and then exchange the authorization grant for an access token. The access token represents the grant's scope, duration, and other attributes granted by the authorization grant. The client accesses the protected resource by presenting the access token to the resource server. In some cases, a client can directly present its own credentials to an authorization server to obtain an access token without having to first obtain an authorization grant from a resource owner.

The access token provides an abstraction, 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.

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |        Authorization Grant &  +---------------+
|        |--(C)--- Client Credentials -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

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

This section defines three methods of sending bearer access tokens in resource requests to resource servers. Clients MUST NOT use more than one method to transmit the token in each request.

2.1. Authorization Request Header Field

TOC

When sending the access token in the `Authorization` request header field defined by HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]**, the client uses the `Bearer` authentication scheme to transmit the access token.

For example:

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

The `Authorization` header field uses the framework defined by HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]** as follows:

```
credentials = "Bearer" 1*SP b64token
```

The b64token syntax was chosen over the alternative #auth-param syntax also defined by HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]** both for simplicity and for compatibility with existing implementations. If additional parameters are needed in the future, a different scheme would need to be defined.

Clients SHOULD make authenticated requests with a bearer token using the `Authorization` request header field with the `Bearer` HTTP authorization scheme. Resource servers MUST support this method.

TOC

2.2. Form-Encoded Body Parameter

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

- The HTTP request entity-header includes the `Content-Type` header field set to `application/x-www-form-urlencoded`.
- The entity-body follows the encoding requirements of the `application/x-www-form-urlencoded` content-type as defined by HTML 4.01 **[W3C.REC-html401-19991224]**.
- The HTTP request entity-body is single-part.
- The content to be encoded in the entity-body MUST consist entirely of ASCII **[USASCII]** characters.
- The HTTP request method is one for which the request body has defined semantics. In particular, this means that the `GET` method MUST NOT be used.

The entity-body MAY include other request-specific parameters, in which case, the `access_token` parameter MUST be properly separated from the request-specific parameters using `&` character(s) (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

access_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. Resource servers MAY support this method.

2.3. URI Query Parameter

When sending the access token in the HTTP request URI, the client adds the access token to the request URI query component as defined by Uniform Resource Identifier (URI) **[RFC3986]** using the `access_token` parameter.

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

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

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

For example:

```
https://server.example.com/resource?x=y&access_token=vF9dft4qmT&p=q
```

Because of the security weaknesses associated with the URI method (see **Section 4**), including the high likelihood that the URL containing the access token will be logged, it SHOULD NOT be used unless it is impossible to transport the access token in the `Authorization` request header field or the HTTP request entity-body. Resource servers MAY support this method.

---

3. The WWW-Authenticate Response Header Field

If the protected resource request does not include authentication credentials or does not contain an access token that enables access to the protected resource, 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 HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]**.

All challenges defined by this specification MUST use the auth-scheme value `Bearer`. This scheme MUST be followed by one or more auth-param values. The auth-param attributes used or defined by this specification are as follows. Other auth-param attributes MAY be used as well.

A `realm` attribute MAY be included to indicate the scope of protection in the manner described in HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]**. The `realm` attribute MUST NOT appear more than once.

The `scope` attribute is a space-delimited list of scope values indicating the required scope of the access token for accessing the requested resource. In some cases, the `scope` value will be used when requesting a new access token with sufficient scope of access to utilize the protected resource. Use of the `scope` attribute is OPTIONAL. The `scope` attribute MUST NOT appear more than once. The `scope` value is intended for programmatic use and is not meant to be displayed to end users.

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 3.1**. In addition, the resource server MAY include the `error_description` attribute to provide developers a human-readable explanation that is not meant to be displayed to end users. It also MAY include the `error_uri` attribute with an absolute URI identifying a human-readable web page explaining the error. The `error`, `error_description`, and `error_uri` attributes MUST NOT appear more than once.

Values for the `scope` attribute MUST NOT include characters outside the set %x21 / %x23-5B / %x5D-7E for representing scope values and %x20 for delimiters between scope values.

Values for the `error` and `error_description` attributes MUST NOT include characters outside the set %x20-21 / %x23-5B / %x5D-7E. Values for the `error_uri` attribute MUST conform to the URI-Reference syntax, and thus MUST NOT include characters outside the set %x21 / %x23-5B / %x5D-7E.

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"
```

3.1. Error Codes

TOC

When a request fails, the resource server responds using the appropriate HTTP status code (typically, 400, 401, 403, or 405), 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.

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"
```

---

4. Security Considerations

TOC

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

4.1. Security Threats

TOC

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 a token generated for consumption by one resource server to gain access to a different resource server that mistakenly believes the token to be for it.

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

4.2. Threat Mitigation

TOC

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. The mechanics of such an interaction are not defined by this specification.

This document does not specify the encoding or the contents of the token; hence detailed recommendations about the means of guaranteeing token integrity protection are outside the scope of this document. The token integrity protection MUST be 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.

The authorization server MUST implement TLS. Which version(s) ought to be implemented will vary over time, and depend on the widespread deployment and known security vulnerabilities at the time of implementation. At the time of this writing, TLS version 1.2 **[RFC5246]** is the most recent version, but has very limited actual deployment, and might not be readily available in implementation toolkits. TLS version 1.0 **[RFC2246]** is the most widely deployed version, and will give the broadest interoperability.

To protect against token disclosure, confidentiality protection MUST be applied using TLS **[RFC5246]** with a ciphersuite that provides confidentiality and integrity 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 and integrity 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 MUST be applied in addition to the usage of TLS protection. As a further defense against token disclosure, the client MUST validate the TLS certificate chain when making requests to protected resources.

Cookies are typically transmitted in the clear. Thus, any information contained in them is at risk of disclosure. Therefore, bearer tokens MUST NOT be stored in cookies that can be sent in the clear.

In some deployments, including those utilizing load balancers, the TLS connection to the

resource server terminates prior to the actual server that provides the resource. This could leave the token unprotected between the front end server where the TLS connection terminates and the back end server that provides the resource. In such deployments, sufficient measures MUST be employed to ensure confidentiality of the token between the front end and back end servers; encryption of the token is one possible such measure.

To deal with token capture and replay, the following recommendations are made: First, the lifetime of the token MUST be limited; one means of achieving this is by putting a validity time field inside the protected part of the token. Note that using short-lived (one hour or less) tokens reduces the impact 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. 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 Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS) **[RFC6125]**. 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

4.3. 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) or equivalent transport security 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). Implementations that do store bearer tokens in cookies MUST take precautions against cross site request forgery.

Issue short-lived bearer tokens:
: Token servers SHOULD issue short-lived (one hour or less) bearer tokens, particularly when issuing tokens to clients that run within a web browser or other environments where information leakage may occur. Using short-lived bearer tokens can reduce the impact of them being leaked.

Issue scoped bearer tokens:
: Token servers SHOULD 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:
: Bearer tokens SHOULD NOT be passed in page URLs (for example as query string parameters). Instead, bearer tokens SHOULD be passed in HTTP message headers or message bodies for which confidentiality measures are taken. 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, attackers might be able to steal them from the history data, logs, or other unsecured locations.

TOC

5. IANA Considerations

5.1. OAuth Access Token Type Registration

TOC

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

5.1.1. 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]]

5.2. Authentication Scheme Registration

TOC

This specification registers the following authentication scheme in the Authentication Scheme Registry defined in HTTP/1.1, Part 7 **[I-D.ietf-httpbis-p7-auth]**.

5.2.1. The "Bearer" Authentication Scheme

TOC

Authentication Scheme Name:
Bearer
Pointer to specification text:
[[this document]]
Notes (optional):
(none)

---

6. References

TOC

6.1. Normative References

TOC

[I-D.ietf-httpbis-p1-messaging] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "**HTTP/1.1, part 1: URIs, Connections, and Message Parsing**," draft-ietf-httpbis-p1-messaging-18 (work in progress), January 2012 (**TXT**).

[I-D.ietf-httpbis-p7-auth] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "**HTTP/1.1, part 7: Authentication**," draft-ietf-httpbis-p7-auth-18 (work in progress), January 2012 (**TXT**).

[I-D.ietf-oauth-v2] Hammer, E., Recordon, D., and D. Hardt, "**The OAuth 2.0 Authorization Protocol**," draft-ietf-oauth-v2-23 (work in progress), January 2012 (**TXT**, **PDF**).

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

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

[RFC6125] Saint-Andre, P. and J. Hodges, "**Representation and Verification of Domain-Based Application Service**

Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)," RFC 6125, March 2011 (TXT).

[USASCII] American National Standards Institute, "Coded Character Set -- 7-bit American Standard Code for Information Interchange," ANSI X3.4, 1986.

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

6.2. Informative References

TOC

[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.

[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).

Appendix A. Acknowledgements

TOC

The following people contributed to preliminary versions of this document: Blaine Cook (BT), Brian Eaton (Google), Yaron Y. 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, Amanda Anganes, Andrew Arnott, Dirk Balfanz, John Bradley, Brian Campbell, Leah Culver, Bill de hÓra, Brian Ellin, Igor Faynberg, Stephen Farrell, George Fletcher, Tim Freeman, Evan Gilbert, Yaron Y. Goland, Thomas Hardjono, Justin Hart, Phil Hunt, John Kemp, Eran Hammer-Lahav, Chasen Le Hara, Barry Leiba, Michael B. Jones, Torsten Lodderstedt, Eve Maler, James Manger, Laurence Miao, William J. Mills, Chuck Mortimore, Anthony Nadalin, Julian Reschke, Justin Richer, Peter Saint-Andre, Nat Sakimura, Rob Sayre, Marius Scurtescu, Naitik Shah, Justin Smith, Jeremy Suriel, Christian Stübner, Paul Tarjan, Hannes Tschofenig, Franklin Tse, and Shane Weeden.

Appendix B. Document History

TOC

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

-16

- Use the HTTPbis auth-param syntax for Bearer challenge attributes.
- Dropped the sentence "The `realm` value is intended for programmatic use and is not meant to be displayed to end users".
- Reordered form-encoded body parameter description bullets for better readability.
- Added **[USASCII]** reference.

-15

- Clarified that form-encoded content must consist entirely of ASCII characters.
- Added TLS version requirements.
- Applied editorial improvements suggested by Mark Nottingham during the APPS area review.

-14

- Changes made in response to review comments by Security Area Director Stephen Farrell. Specifically:
- Strengthened warnings about passing an access token as a query parameter and more precisely described the limitations placed upon the use of this method.
- Clarified that the `realm` attribute MAY included to indicate the scope of protection in the manner described in HTTP/1.1, Part 7

[I-D.ietf-httpbis-p7-auth].

- Normatively stated that "the token integrity protection MUST be sufficient to prevent the token from being modified".
- Added statement that "TLS is mandatory to implement and use with this specification" to the introduction.
- Stated that TLS MUST be used with "a ciphersuite that provides confidentiality and integrity protection".
- Added "As a further defense against token disclosure, the client MUST validate the TLS certificate chain when making requests to protected resources" to the Threat Mitigation section.
- Clarified that putting a validity time field inside the protected part of the token is one means, but not the only means, of limiting the lifetime of the token.
- Dropped the confusing phrase "for instance, through the use of TLS" from the sentence about confidentiality protection of the exchanges.
- Reference RFC 6125 for identity verification, rather than RFC 2818.
- Stated that the token MUST be protected between front end and back end servers when the TLS connection terminates at a front end server that is distinct from the actual server that provides the resource.
- Stated that bearer tokens MUST NOT be stored in cookies that can be sent in the clear in the Threat Mitigation section.
- Replaced sole remaining reference to **[RFC2616]** with HTTPbis **[I-D.ietf-httpbis-p1-messaging]** reference.
- Replaced all references where the reference is used as if it were part of the sentence (such as "defined by [I-D.whatever]") with ones where the specification name is used, followed by the reference (such as "defined by Whatever [I-D.whatever]").
- Other on-normative editorial improvements.

-13

- At the request of Hannes Tschofenig, made ABNF changes to make it clear that no special WWW-Authenticate response header field parsers are needed. The `scope`, `error-description`, and `error-uri` parameters are all now defined as quoted-string in the ABNF (as `error` already was). Restrictions on these values that were formerly described in the ABNFs are now described in normative text instead.

-12

- Made non-normative editorial changes that Hannes Tschofenig requested be applied prior to forwarding the specification to the IESG.
- Added rationale for the choice of the b64token syntax.
- Added rationale stating that receivers are free to parse the `scope` attribute using a standard quoted-string parser, since it will correctly process all legal `scope` values.
- Added additional active working group contributors to the Acknowledgements section.

-11

- Replaced uses of <"> with DQUOTE to pass ABNF syntax check.

-10

- Removed the #auth-param option from Authorization header syntax (leaving only the b64token syntax).
- Restricted the `scope` value character set to %x21 / %x23-5B / %x5D-7E (printable ASCII characters excluding double-quote and backslash). Indicated that scope is intended for programmatic use and is not meant to be displayed to end users.
- Restricted the character set for `error_description` strings to SP / VCHAR and indicated that they are not meant to be displayed to end users.
- Included more description in the Abstract, since Hannes Tschofenig indicated that the RFC editor would require this.
- Changed "Access Grant" to "Authorization Grant", as was done in the core spec.
- Simplified the introduction to the Authenticated Requests section.

-09

- Incorporated working group last call comments. Specific changes were:
- Use definitions from **[I-D.ietf-httpbis-p7-auth]** rather than **[RFC2617]**.
- Update credentials definition to conform to **[I-D.ietf-httpbis-p7-auth]**.
- Further clarified that query parameters may occur in any order.
- Specify that error_description is UTF-8 encoded (matching the core specification).
- Registered "Bearer" Authentication Scheme in Authentication Scheme Registry defined by **[I-D.ietf-httpbis-p7-auth]**.
- Updated references to oauth-v2, httpbis-p1-messaging, and httpbis-p7-auth drafts.
- Other wording improvements not introducing normative changes.

-08

- Updated references to oauth-v2 and HTTPbis drafts.

-07

- Added missing comma in error response example.

-06

- Changed parameter name `bearer_token` to `access_token`, per working group consensus.
- Changed HTTP status code for `invalid_request` error code from HTTP 401 (Unauthorized) back to HTTP 400 (Bad Request), per input from HTTP working group experts.

-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/