OAuth Working Group M. Jones
Internet-Draft Microsoft
Intended status: Standards Track J. Bradley
Expires: March 24, 2017 B. Campbell
Ping Identity
September 20, 2016

OAuth 2.0 Token Binding
draft-ietf-oauth-token-binding-01

Abstract

This specification enables OAuth 2.0 implementations to apply Token Binding to Access Tokens and Refresh Tokens. This cryptographically binds these tokens to the TLS connections over which they are intended to be used. This use of Token Binding protects these tokens from man-in-the-middle and token export and replay attacks.

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 March 24, 2017.

Copyright Notice

Copyright (c) 2016 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
  1.1. Requirements Notation and Conventions 3
  1.2. Terminology 3
2. Token Binding for Refresh Tokens 3
3. Token Binding for Access Tokens 4
  3.1. Access Tokens Issued from the Authorization Endpoint . . 5
  3.2. Access Tokens Issued from the Token Endpoint 5
  3.3. Protected Resource Token Binding Validation 5
  3.4. Representing Token Binding in JWT Access Tokens 5
4. Phasing in Token Binding and Preventing Downgrade Attacks . . 6
5. Token Binding Metadata 7
  5.1. Token Binding Client Metadata 7
  5.2. Token Binding Authorization Server Metadata 7
  5.3. Token Binding Protected Resource Metadata 7
6. Security Considerations 8
7. IANA Considerations 8
  7.1. OAuth Dynamic Client Registration Metadata Registration . 8
    7.1.1. Registry Contents 8
  7.2. OAuth Authorization Server Metadata Registration 8
    7.2.1. Registry Contents 9
  7.3. OAuth Protected Resource Metadata Registration 9
    7.3.1. Registry Contents 9
8. References 9
  8.1. Normative References 9
  8.2. Informative References 11
Appendix A. Acknowledgements 11
Appendix B. Open Issues 11
Appendix C. Document History 11
Authors' Addresses 12

1. Introduction

This specification enables OAuth 2.0 [RFC6749] implementations to apply Token Binding The Token Binding Protocol Version 1.0 [I-D.ietf-tokbind-protocol] Token Binding over HTTP [I-D.ietf-tokbind-https] to Access Tokens and Refresh Tokens. This cryptographically binds these tokens to the TLS connections over which they are intended to be used. This use of Token Binding protects these tokens from man-in-the-middle and token export and replay attacks.

1.1. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology

This specification uses the terms "Access Token", "Authorization Endpoint", "Authorization Server", "Client", "Protected Resource", "Refresh Token", and "Token Endpoint" defined by OAuth 2.0 [RFC6749], the terms "Claim" and "JSON Web Token (JWT)" defined by JSON Web Token (JWT) [JWT], the term "User Agent" defined by RFC 7230 [RFC7230], and the terms "Provided", "Referred", "Token Binding" and "Token Binding ID" defined by Token Binding over HTTP [I-D.ietf-tokbind-https].

2. Token Binding for Refresh Tokens

Token Binding of refresh tokens is a straightforward first-party scenario, applying term "first-party" as used in Token Binding over HTTP [I-D.ietf-tokbind-https]. It cryptographically binds the refresh token to the TLS connection between the client and the token endpoint. This case is straightforward because the refresh token is both retrieved by the client from the token endpoint and sent by the client to the token endpoint. Unlike the federated scenarios described in Section 4 (Federation Use Cases) of Token Binding over HTTP [I-D.ietf-tokbind-https] and the access token case described in the next section, only a single TLS connection is involved in the refresh token case.

Token Binding a refresh token requires that the authorization server do two things. First, when refresh token is sent to the client, the authorization server needs to remember the Provided Token Binding ID and remember its association with the issued refresh token. Second, when a token request containing a refresh token is received at the token endpoint, the authorization server needs to verify that the Provided Token Binding ID for the request matches the remembered Token Binding ID associated with the refresh token. If the Token Binding IDs do not match, the authorization server should return an error in response to the request.

How the authorization server remembers the association between the refresh token and the Token Binding ID is an implementation detail that beyond the scope of this specification. Some authorization servers will choose to store the Token Binding ID (or a cryptographic hash of it, such a SHA-256 hash [SHS]) in the refresh token itself,

thus reducing the amount of state to be kept by the server. Other authorization servers will add the Token Binding ID value (or a hash of it) to an internal data structure also containing other information about the refresh token, such as grant type information. These choices make no difference to the client, since the refresh token is opaque to it.

3. Token Binding for Access Tokens

Token Binding for access tokens cryptographically binds the access token to the TLS connection between the client and the protected resource. Token Binding is applied to access tokens in a similar manner to that described in Section 4 (Federation Use Cases) of Token Binding over HTTP [I-D.ietf-tokbind-https]. It also builds upon the mechanisms for Token Binding of ID Tokens defined in OpenID Connect Token Bound Authentication 1.0 [OpenID.TokenBinding].

In the OpenID Connect [OpenID.Core] use case, HTTP redirects are used to pass information between the identity provider and the relying party; this HTTP redirect makes the Token Binding ID of the relying party available to the identity provider as the Referred Token Binding ID, information about which is then added to the ID Token. No such redirect occurs between the authorization server and the protected resource in the access token case; therefore, information about the Token Binding ID for the TLS connection between the client and the protected resource needs to be explicitly communicated by the client to the authorization server to achieve Token Binding of the access token.

This information is passed to the authorization server using the Referred Token Binding ID, just as in the ID Token case. The only difference is that the client needs to explicitly communicate the Token Binding ID of the TLS connection between the client and the protected resource to the Token Binding implementation so that it is sent as the Referred Token Binding ID in the request to the authorization server. This functionality provided by Token Binding implementations is described in Section 5 (Implementation Considerations) of Token Binding over HTTP [I-D.ietf-tokbind-https].

Note that to obtain this Token Binding ID, the client may need to establish a TLS connection between itself and the protected resource prior to making the request to the authorization server so that the Provided Token Binding ID for the TLS connection to the protected resource can be obtained. How the client retrieves this Token Binding ID from the underlying Token Binding API is implementation and operating system specific. An alternative, if supported, is for the client to generate a Token Binding key to use for the protected resource, use the Token Binding ID for that key, and then later use

that key when the TLS connection to the protected resource is established.

3.1. Access Tokens Issued from the Authorization Endpoint

For access tokens returned directly from the authorization endpoint, such as with the implicit grant defined in Section 4.2 of OAuth 2.0 [RFC6749], the Referred Token Binding ID used to bind the access token is sent with the authorization request. Upon receiving the Referred Token Binding ID in an authorization request, the authorization server then records it (or a cryptographic hash of it) in the issued access token. Alternatively, in some implementations, the resource's Token Binding ID information might be communicated to the protected resource by other means, such as by introspecting [RFC7662] the access token.

3.2. Access Tokens Issued from the Token Endpoint

For access tokens returned from the token endpoint, the Referred Token Binding ID used to bind the access token is sent with the token request. This applies to all the conventional grant types from OAuth 2.0 [RFC6749], including but not limited to refresh and authorization code token requests, as well as extension grants, such as JWT assertion authorization grants [RFC7523]. In this case, the Token Binding ID of the TLS connection between the client and the protected resource is sent to the authorization server at the token endpoint as the Referred Token Binding ID. The authorization server then records it (or a cryptographic hash of it) in the issued access token or communicates it to the protected resource by other means, just as in the previous case.

3.3. Protected Resource Token Binding Validation

Upon receiving a token bound access token, the protected resource validates the binding by comparing the Provided Token Binding ID to the Token Binding ID for the access token. Alternatively, cryptographic hashes of these Token Binding ID values can be compared. If the values do not match, the resource access attempt MUST be rejected with an error.

3.4. Representing Token Binding in JWT Access Tokens

If the access token is represented as a JWT, the token binding information SHOULD be represented in the same way that it is in token bound OpenID Connect ID Tokens [OpenID.TokenBinding]. That specification defines the new JWT Confirmation Method RFC 7800 [RFC7800] member "tbh" (token binding hash) to represent the SHA-256 hash of a Token Binding ID in an ID Token. The value of the "tbh"

member is the base64url encoding of the SHA-256 hash of the Token Binding ID.

The following example demonstrates the JWT Claims Set of an access token containing the base64url encoding of the SHA-256 hash of a Token Binding ID as the value of the "tbh" (token binding hash) element in the "cnf" (confirmation) claim:

```
  {
   "iss": "https://server.example.com",
   "aud": "https://resource.example.com",
   "iat": 1467324320,
   "exp": 1467324920,
   "cnf":{
     "tbh": "n0jI3trBK6_Gp2qiLOf48ZEZTjpBnhm-QOyzJxhBeAk"
    }
  }
```

4. Phasing in Token Binding and Preventing Downgrade Attacks

Many OAuth implementations will be deployed in situations in which not all participants support Token Binding. Any of combination of the client, the authorization server, the protected resource, and the user agent may not yet support Token Binding, in which case it will not work end-to-end.

It is a context-dependent deployment choice whether to allow interactions to proceed in which Token Binding is not supported or whether to treat Token Binding failures at any step as fatal errors. Particularly in dynamic deployment environments in which End Users have choices of clients, authorization servers, protected resources, and/or user agents, it is RECOMMENDED that authorizations using one or more components that do not implement Token Binding be allowed to successfully proceed. This enables different components to be upgraded to supporting Token Binding at different times, providing a smooth transition path for phasing in Token Binding. However, when Token Binding has been performed, any Token Binding key mismatches MUST be treated as fatal errors.

If all the participants in an authorization interaction support Token Binding and yet one or more of them does not use it, this is likely evidence of a downgrade attack. In this case, the authorization SHOULD be aborted with an error. For instance, if the protected resource knows that the authorization server and the user agent both support Token Binding and yet the access token received does not contain Token Binding information, this is almost certainly a sign of an attack.

The authorization server, client, and protected resource can determine whether the others support Token Binding using the metadata values defined in the next section. They can determine whether the user agent supports Token Binding by whether it negotiated Token Binding for the TLS connection.

5. Token Binding Metadata

5.1. Token Binding Client Metadata

Clients supporting Token Binding that also support the OAuth 2.0 Dynamic Client Registration Protocol [RFC7591] use these metadata values to declare their support for Token Binding of access tokens and refresh tokens:

client_access_token_token_binding_supported
OPTIONAL. Boolean value specifying whether the client supports Token Binding of access tokens. If omitted, the default value is "false".

client_refresh_token_token_binding_supported
OPTIONAL. Boolean value specifying whether the client supports Token Binding of refresh tokens. If omitted, the default value is "false".

5.2. Token Binding Authorization Server Metadata

Authorization servers supporting Token Binding that also support OAuth 2.0 Authorization Server Metadata [OAuth.AuthorizationMetadata] use these metadata values to declare their support for Token Binding of access tokens and refresh tokens:

as_access_token_token_binding_supported
OPTIONAL. Boolean value specifying whether the authorization server supports Token Binding of access tokens. If omitted, the default value is "false".

as_refresh_token_token_binding_supported
OPTIONAL. Boolean value specifying whether the authorization server supports Token Binding of refresh tokens. If omitted, the default value is "false".

5.3. Token Binding Protected Resource Metadata

Protected resources supporting Token Binding that also support the OAuth 2.0 Protected Resource Metadata [OAuth.ResourceMetadata] use this metadata value to declare their support for Token Binding of access tokens:

resource_access_token_token_binding_supported
OPTIONAL. Boolean value specifying whether the protected resource supports Token Binding of access tokens. If omitted, the default value is "false".

6. Security Considerations

If a refresh request is received by the authorization server containing a Referred Token Binding ID and the refresh token in the request is not itself token bound, then it is not clear that token binding the access token adds significant value. This situation should be considered an open issue for discussion by the working group.

7. IANA Considerations

7.1. OAuth Dynamic Client Registration Metadata Registration

This specification registers the following client metadata definitions in the IANA "OAuth Dynamic Client Registration Metadata" registry [IANA.OAuth.Parameters] established by [RFC7591]:

7.1.1. Registry Contents

- Client Metadata Name: "client_access_token_token_binding_supported"
- Client Metadata Description: Boolean value specifying whether the client supports Token Binding of access tokens
- Change Controller: IESG
- Specification Document(s): Section 5.1 of [[this specification]]

- Client Metadata Name: "client_refresh_token_token_binding_supported"
- Client Metadata Description: Boolean value specifying whether the client supports Token Binding of refresh tokens
- Change Controller: IESG
- Specification Document(s): Section 5.1 of [[this specification]]

7.2. OAuth Authorization Server Metadata Registration

This specification registers the following metadata definitions in the IANA "OAuth Authorization Server Metadata" registry established by [OAuth.AuthorizationMetadata]:

7.2.1. Registry Contents

- Metadata Name: "as_access_token_token_binding_supported"
- Metadata Description: Boolean value specifying whether the authorization server supports Token Binding of access tokens
- Change Controller: IESG
- Specification Document(s): Section 5.2 of [[this specification]]

- Metadata Name: "as_refresh_token_token_binding_supported"
- Metadata Description: Boolean value specifying whether the authorization server supports Token Binding of refresh tokens
- Change Controller: IESG
- Specification Document(s): Section 5.2 of [[this specification]]

7.3. OAuth Protected Resource Metadata Registration

This specification registers the following client metadata definition in the IANA "OAuth Protected Resource Metadata" registry [IANA.OAuth.Parameters] established by [OAuth.ResourceMetadata]:

7.3.1. Registry Contents

- Resource Metadata Name: "resource_access_token_token_binding_supported"
- Resource Metadata Description: Boolean value specifying whether the protected resource supports Token Binding of access tokens
- Change Controller: IESG
- Specification Document(s): Section 5.3 of [[this specification]]

8. References

8.1. Normative References

[I-D.ietf-tokbind-https]
Popov, A., Nystrom, M., Balfanz, D., Langley, A., and J. Hodges, "Token Binding over HTTP", draft-ietf-tokbind-https-06 (work in progress), August 2016.

[I-D.ietf-tokbind-protocol]
Popov, A., Nystrom, M., Balfanz, D., Langley, A., and J. Hodges, "The Token Binding Protocol Version 1.0", draft-ietf-tokbind-protocol-10 (work in progress), September 2016.

[IANA.OAuth.Parameters]
IANA, "OAuth Parameters", <http://www.iana.org/assignments/oauth-parameters>.

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <http://tools.ietf.org/html/rfc7519>.

[OAuth.AuthorizationMetadata] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", draft-ietf-oauth-discovery-04 (work in progress), August 2016, <http://tools.ietf.org/html/draft-ietf-oauth-discovery-04>.

[OAuth.ResourceMetadata] Jones, M. and P. Hunt, "OAuth 2.0 Protected Resource Metadata", draft-jones-oauth-resource-metadata-00 (work in progress), August 2016, <http://tools.ietf.org/html/draft-jones-oauth-resource-metadata-00>.

[OpenID.TokenBinding] Jones, M., Bradley, J., and B. Campbell, "OpenID Connect Token Bound Authentication 1.0", July 2016, <http://openid.net/specs/openid-connect-token-bound-authentication-1_0.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <http://www.rfc-editor.org/info/rfc6749>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <http://www.rfc-editor.org/info/rfc7591>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <http://www.rfc-editor.org/info/rfc7662>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <http://www.rfc-editor.org/info/rfc7800>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

8.2. Informative References

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <http://www.rfc-editor.org/info/rfc7523>.

Appendix A. Acknowledgements

The authors would like to thank the following people for their contributions to the specification: Dirk Balfanz, William Denniss, Andrei Popov, and Nat Sakimura.

Appendix B. Open Issues

- What should we do in the case that a refresh request for a token bound access token is received when the refresh token used in the request is not token bound?

Appendix C. Document History

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

-01

- Changed Token Binding for access tokens to use the Referred Token Binding ID, now that the Implementation Considerations in the Token Binding HTTPS specification make it clear that implementations will enable using the Referred Token Binding ID.
- Defined Protected Resource Metadata value.
- Changed to use the more specific term "protected resource" instead of "resource server".

-00

- Created the initial working group version from draft-jones-oauth-token-binding-00.

Authors' Addresses

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/

John Bradley
Ping Identity

Email: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/

Brian Campbell
Ping Identity

Email: brian.d.campbell@gmail.com
URI: https://twitter.com/__b_c