Internet Engineering Task Force (IETF)
Request for Comments: 7523
Category: Standards Track
ISSN: 2070-1721

M. Jones
Microsoft
B. Campbell
Ping Identity
C. Mortimore
Salesforce
May 2015

JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants

Abstract

This specification defines the use of a JSON Web Token (JWT) Bearer Token as a means for requesting an OAuth 2.0 access token as well as for client authentication.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7523.

Table of Contents

1. Introduction

JSON Web Token (JWT) [JWT] is a JSON-based [RFC7159] security token encoding that enables identity and security information to be shared across security domains. A security token is generally issued by an Identity Provider and consumed by a Relying Party that relies on its content to identify the token's subject for security-related purposes.

The OAuth 2.0 Authorization Framework [RFC6749] provides a method for making authenticated HTTP requests to a resource using an access token. Access tokens are issued to third-party clients by an authorization server (AS) with the (sometimes implicit) approval of the resource owner. In OAuth, an authorization grant is an abstract term used to describe intermediate credentials that represent the resource owner authorization. An authorization grant is used by the client to obtain an access token. Several authorization grant types are defined to support a wide range of client types and user experiences. OAuth also allows for the definition of new extension grant types to support additional clients or to provide a bridge between OAuth and other trust frameworks. Finally, OAuth allows the

definition of additional authentication mechanisms to be used by clients when interacting with the authorization server.

"Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7521] is an abstract extension to OAuth 2.0 that provides a general framework for the use of assertions (a.k.a. security tokens) as client credentials and/or authorization grants with OAuth 2.0. This specification profiles the OAuth Assertion Framework [RFC7521] to define an extension grant type that uses a JWT Bearer Token to request an OAuth 2.0 access token as well as for use as client credentials. The format and processing rules for the JWT defined in this specification are intentionally similar, though not identical, to those in the closely related specification "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7522]. The differences arise where the structure and semantics of JWTs differ from SAML Assertions. JWTs, for example, have no direct equivalent to the <SubjectConfirmation> or <AuthnStatement> elements of SAML Assertions.

This document defines how a JWT Bearer Token can be used to request an access token when a client wishes to utilize an existing trust relationship, expressed through the semantics of the JWT, without a direct user-approval step at the authorization server. It also defines how a JWT can be used as a client authentication mechanism. The use of a security token for client authentication is orthogonal to and separable from using a security token as an authorization grant. They can be used either in combination or separately. Client authentication using a JWT is nothing more than an alternative way for a client to authenticate to the token endpoint and must be used in conjunction with some grant type to form a complete and meaningful protocol request. JWT authorization grants may be used with or without client authentication or identification. Whether or not client authentication is needed in conjunction with a JWT authorization grant, as well as the supported types of client authentication, are policy decisions at the discretion of the authorization server.

The process by which the client obtains the JWT, prior to exchanging it with the authorization server or using it for client authentication, is out of scope.

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 RFC 2119 [RFC2119].

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

1.2. Terminology

All terms are as defined in the following specifications: "The OAuth 2.0 Authorization Framework" [RFC6749], the OAuth Assertion Framework [RFC7521], and "JSON Web Token (JWT)" [JWT].

2. HTTP Parameter Bindings for Transporting Assertions

The OAuth Assertion Framework [RFC7521] defines generic HTTP parameters for transporting assertions (a.k.a. security tokens) during interactions with a token endpoint. This section defines specific parameters and treatments of those parameters for use with JWT Bearer Tokens.

2.1. Using JWTs as Authorization Grants

To use a Bearer JWT as an authorization grant, the client uses an access token request as defined in Section 4 of the OAuth Assertion Framework [RFC7521] with the following specific parameter values and encodings.

The value of the "grant_type" is "urn:ietf:params:oauth:grant-type:jwt-bearer".

The value of the "assertion" parameter MUST contain a single JWT.

The "scope" parameter may be used, as defined in the OAuth Assertion Framework [RFC7521], to indicate the requested scope.

Authentication of the client is optional, as described in Section 3.2.1 of OAuth 2.0 [RFC6749] and consequently, the "client_id" is only needed when a form of client authentication that relies on the parameter is used.

The following example demonstrates an access token request with a JWT as an authorization grant (with extra line breaks for display purposes only):

```
POST /token.oauth2 HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
&assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6IjE2In0.
eyJpc3Mi[...omitted for brevity...].
J9l-ZhwP[...omitted for brevity...]
```

2.2. Using JWTs for Client Authentication

To use a JWT Bearer Token for client authentication, the client uses the following parameter values and encodings.

The value of the "client_assertion_type" is "urn:ietf:params:oauth:client-assertion-type:jwt-bearer".

The value of the "client_assertion" parameter contains a single JWT. It MUST NOT contain more than one JWT.

The following example demonstrates client authentication using a JWT during the presentation of an authorization code grant in an access token request (with extra line breaks for display purposes only):

```
POST /token.oauth2 HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
eyJpc3Mi[...omitted for brevity...].
cC4hiUPo[...omitted for brevity...]
```

3. JWT Format and Processing Requirements

In order to issue an access token response as described in OAuth 2.0 [RFC6749] or to rely on a JWT for client authentication, the authorization server MUST validate the JWT according to the criteria below. Application of additional restrictions and policy are at the discretion of the authorization server.

1. The JWT MUST contain an "iss" (issuer) claim that contains a unique identifier for the entity that issued the JWT. In the absence of an application profile specifying otherwise, compliant applications MUST compare issuer values using the Simple String Comparison method defined in Section 6.2.1 of RFC 3986 [RFC3986].

2. The JWT MUST contain a "sub" (subject) claim identifying the principal that is the subject of the JWT. Two cases need to be differentiated:

   A. For the authorization grant, the subject typically identifies an authorized accessor for which the access token is being requested (i.e., the resource owner or an authorized delegate), but in some cases, may be a pseudonymous identifier or other value denoting an anonymous user.

   B. For client authentication, the subject MUST be the "client_id" of the OAuth client.

3. The JWT MUST contain an "aud" (audience) claim containing a value that identifies the authorization server as an intended audience. The token endpoint URL of the authorization server MAY be used as a value for an "aud" element to identify the authorization server as an intended audience of the JWT. The authorization server MUST reject any JWT that does not contain its own identity as the intended audience. In the absence of an application profile specifying otherwise, compliant applications MUST compare the audience values using the Simple String Comparison method defined in Section 6.2.1 of RFC 3986 [RFC3986]. As noted in Section 5, the precise strings to be used as the audience for a given authorization server must be configured out of band by the authorization server and the issuer of the JWT.

4. The JWT MUST contain an "exp" (expiration time) claim that limits the time window during which the JWT can be used. The authorization server MUST reject any JWT with an expiration time that has passed, subject to allowable clock skew between systems. Note that the authorization server may reject JWTs with an "exp" claim value that is unreasonably far in the future.

5. The JWT MAY contain an "nbf" (not before) claim that identifies the time before which the token MUST NOT be accepted for processing.

6. The JWT MAY contain an "iat" (issued at) claim that identifies the time at which the JWT was issued. Note that the authorization server may reject JWTs with an "iat" claim value that is unreasonably far in the past.

7. The JWT MAY contain a "jti" (JWT ID) claim that provides a unique identifier for the token. The authorization server MAY ensure that JWTs are not replayed by maintaining the set of used "jti" values for the length of time for which the JWT would be considered valid based on the applicable "exp" instant.

8. The JWT MAY contain other claims.

9. The JWT MUST be digitally signed or have a Message Authentication Code (MAC) applied by the issuer. The authorization server MUST reject JWTs with an invalid signature or MAC.

10. The authorization server MUST reject a JWT that is not valid in all other respects per "JSON Web Token (JWT)" [JWT].

3.1. Authorization Grant Processing

JWT authorization grants may be used with or without client authentication or identification. Whether or not client authentication is needed in conjunction with a JWT authorization grant, as well as the supported types of client authentication, are policy decisions at the discretion of the authorization server. However, if client credentials are present in the request, the authorization server MUST validate them.

If the JWT is not valid, or the current time is not within the token's valid time window for use, the authorization server constructs an error response as defined in OAuth 2.0 [RFC6749]. The value of the "error" parameter MUST be the "invalid_grant" error code. The authorization server MAY include additional information regarding the reasons the JWT was considered invalid using the "error_description" or "error_uri" parameters.

For example:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
 "error":"invalid_grant",
 "error_description":"Audience validation failed"
}
```

3.2. Client Authentication Processing

If the client JWT is not valid, the authorization server constructs an error response as defined in OAuth 2.0 [RFC6749]. The value of the "error" parameter MUST be the "invalid_client" error code. The authorization server MAY include additional information regarding the reasons the JWT was considered invalid using the "error_description" or "error_uri" parameters.

4. Authorization Grant Example

The following examples illustrate what a conforming JWT and an access token request would look like.

The example shows a JWT issued and signed by the system entity identified as "https://jwt-idp.example.com". The subject of the JWT is identified by email address as "mike@example.com". The intended audience of the JWT is "https://jwt-rp.example.net", which is an identifier with which the authorization server identifies itself. The JWT is sent as part of an access token request to the authorization server's token endpoint at "https://authz.example.net/token.oauth2".

Below is an example JSON object that could be encoded to produce the JWT Claims Set for a JWT:

```
{"iss":"https://jwt-idp.example.com",
 "sub":"mailto:mike@example.com",
 "aud":"https://jwt-rp.example.net",
 "nbf":1300815780,
 "exp":1300819380,
 "http://claims.example.com/member":true}
```

The following example JSON object, used as the header of a JWT, declares that the JWT is signed with the Elliptic Curve Digital Signature Algorithm (ECDSA) P-256 SHA-256 using a key identified by the "kid" value "16".

```
{"alg":"ES256","kid":"16"}
```

To present the JWT with the claims and header shown in the previous example as part of an access token request, for example, the client might make the following HTTPS request (with extra line breaks for display purposes only):

```
POST /token.oauth2 HTTP/1.1
Host: authz.example.net
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
&assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6IjE2In0.
eyJpc3Mi[...omitted for brevity...].
J9l-ZhwP[...omitted for brevity...]
```

5. Interoperability Considerations

Agreement between system entities regarding identifiers, keys, and endpoints is required in order to achieve interoperable deployments of this profile. Specific items that require agreement are as follows: values for the issuer and audience identifiers, the location of the token endpoint, the key used to apply and verify the digital signature or MAC over the JWT, one-time use restrictions on the JWT, maximum JWT lifetime allowed, and the specific subject and claim requirements of the JWT. The exchange of such information is explicitly out of scope for this specification. In some cases, additional profiles may be created that constrain or prescribe these values or specify how they are to be exchanged. Examples of such profiles include the OAuth 2.0 Dynamic Client Registration Core Protocol [OAUTH-DYN-REG], OpenID Connect Dynamic Client Registration 1.0 [OpenID.Registration], and OpenID Connect Discovery 1.0 [OpenID.Discovery].

The "RS256" algorithm, from [JWA], is a mandatory-to-implement JSON Web Signature algorithm for this profile.

6. Security Considerations

The security considerations described within the following specifications are all applicable to this document: "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7521], "The OAuth 2.0 Authorization Framework" [RFC6749], and "JSON Web Token (JWT)" [JWT].

The specification does not mandate replay protection for the JWT usage for either the authorization grant or for client authentication. It is an optional feature, which implementations may employ at their own discretion.

7. Privacy Considerations

A JWT may contain privacy-sensitive information and, to prevent disclosure of such information to unintended parties, should only be transmitted over encrypted channels, such as Transport Layer Security (TLS). In cases where it is desirable to prevent disclosure of certain information to the client, the JWT should be encrypted to the authorization server.

Deployments should determine the minimum amount of information necessary to complete the exchange and include only such claims in the JWT. In some cases, the "sub" (subject) claim can be a value representing an anonymous or pseudonymous user, as described in Section 6.3.1 of the OAuth Assertion Framework [RFC7521].

8. IANA Considerations

8.1. Sub-Namespace Registration of urn:ietf:params:oauth:grant-type:jwt-bearer

This section registers the value "grant-type:jwt-bearer" in the IANA "OAuth URI" registry established by "An IETF URN Sub-Namespace for OAuth" [RFC6755].

- URN: urn:ietf:params:oauth:grant-type:jwt-bearer
- Common Name: JWT Bearer Token Grant Type Profile for OAuth 2.0
- Change Controller: IESG
- Specification Document: RFC 7523

8.2. Sub-Namespace Registration of urn:ietf:params:oauth:client-assertion-type:jwt-bearer

This section registers the value "client-assertion-type:jwt-bearer" in the IANA "OAuth URI" registry established by "An IETF URN Sub-Namespace for OAuth" [RFC6755].

- URN: urn:ietf:params:oauth:client-assertion-type:jwt-bearer
- Common Name: JWT Bearer Token Profile for OAuth 2.0 Client Authentication
- Change Controller: IESG
- Specification Document: RFC 7523

9. References

9.1. Normative References

[JWA] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <http://www.rfc-editor.org/info/rfc7518>.

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

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <http://www.rfc-editor.org/info/rfc7521>.

9.2. Informative References

[OAUTH-DYN-REG] Richer, J., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", Work in Progress, draft-ietf-oauth-dyn-reg-29, May 2015.

[OpenID.Discovery] Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-discovery-1_0.html>.

[OpenID.Registration]
Sakimura, N., Bradley, J., and M. Jones, "OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-registration-1_0.html>.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, DOI 10.17487/RFC6755, October 2012, <http://www.rfc-editor.org/info/rfc6755>.

[RFC7522] Campbell, B., Mortimore, C., and M. Jones, "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7522, DOI 10.17487/RFC7522, May 2015, <http://www.rfc-editor.org/info/rfc7522>.

Acknowledgements

This profile was derived from "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7522], which has the same authors as this document.

Authors' Addresses

Michael B. Jones
Microsoft

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

Brian Campbell
Ping Identity

EMail: brian.d.campbell@gmail.com

Chuck Mortimore
Salesforce

EMail: cmortimore@salesforce.com