Network Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	B. Campbell
Expires: June 15, 2012	Ping Identity Corp.
	C. Mortimore
	Salesforce.com
	December 13, 2011

TOC

JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0
draft-jones-oauth-jwt-bearer-03

Abstract

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”

This Internet-Draft will expire on June 15, 2012.

Table of Contents

TOC

1. Introduction

JSON Web Token (JWT) [JWT] is a JSON-based security token encoding that enables identity and security information to be shared across security domains. JWTs utilize JSON data structures, as defined in **RFC 4627** [RFC4627]. 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 Protocol [I-D.ietf.oauth-v2] 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.

The **OAuth 2.0 Assertion Profile** [I-D.ietf.oauth-assertions] 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 2.0 Assertion Profile** [I-D.ietf.oauth-assertions] to define an extension grant type that uses a JSON Web Token (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 **SAML 2.0 Bearer Assertion Profiles for OAuth 2.0** [I-D.ietf-oauth-saml2-bearer].

This document defines how a JSON Web Token (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 (and digital signature calculated over) 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 and separable from using a security token as an authorization grant and the two can be used either in combination or in isolation.

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.

TOC

1.1. Notational Conventions

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

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

1.2. Terminology

TOC

All terms are as defined in **The OAuth 2.0 Authorization Protocol** [I-D.ietf.oauth-v2], **OAuth 2.0 Assertion Profile** [I-D.ietf.oauth-assertions], and **JSON Web Token (JWT)** [JWT].

2. HTTP Parameter Bindings for Transporting Assertions

TOC

The **OAuth 2.0 Assertion Profile** [I-D.ietf.oauth-assertions] defines generic HTTP parameters for transporting Assertions (a.k.a. Security Tokens) during interactions with a token endpoint. This section defines the values of those parameters for use with JWT Bearer Tokens.

2.1. Using JWTs as Authorization Grants

TOC

To use a JWT Bearer Token as an authorization grant, use the following parameter values and encodings.

The value of "grant_type" parameter MUST be "urn:ietf:params:oauth:grant-type:jwt-bearer".

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

2.2. Using JWTs for Client Authentication

TOC

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

The value of "client_assertion_type" parameter MUST be "urn:ietf:params:oauth:client-assertion-type:jwt-bearer".

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

3. JWT Format and Processing Requirements

TOC

In order to issue an access token response as described in **The OAuth 2.0 Authorization Protocol** [I-D.ietf.oauth-v2] 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.

- The JWT MUST contain an `iss` (issuer) claim that contains a unique identifier for the entity that issued the JWT.
- The JWT MUST contain a `prn` (principal) claim identifying the subject of the transaction. The principal MAY identify the resource owner for whom the access token is being requested. For client authentication, the principal MUST be the client_id of the OAuth client. When using JWTs as an authorization grant, the principal SHOULD identify an authorized accessor for whom the access token is being requested (typically the resource owner, or an authorized delegate).
- The JWT MUST contain an `aud` (audience) claim containing a URI reference that identifies the authorization server, or the service provider principal entity of its controlling domain, as an intended audience. The token endpoint URL of the authorization server MAY be used as an acceptable value for an `aud` element. The authorization server MUST verify that it is an intended audience for the JWT.
- The JWT MUST contain an `exp` (expiration) claim that limits the time window during which the JWT can be used. The authorization server MUST verify that the expiration time has not passed, subject to allowable clock skew between systems. The authorization server MAY reject JWTs with an `exp` claim value that is

unreasonably far in the future.

- The JWT MAY contain an `nbf` (not before) claim that identifies the time before which the token MUST NOT be accepted for processing.
- 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.
- The JWT MAY contain other claims.
- The JWT MUST be digitally signed by the issuer and the authorization server MUST verify the signature.
- The authorization server MUST verify that the JWT is valid in all other respects per **JSON Web Token (JWT)** [JWT].

TOC

3.1. Authorization Grant Processing

If present, the authorization server MUST also validate the client credentials.

Authorization servers SHOULD issue access tokens with a limited lifetime and require clients to refresh them by requesting a new access token using the same JWT, if it is still valid, or with a new JWT. The authorization server SHOULD NOT issue a refresh token.

If the JWT is not valid, or the current time is not within the token's valid time window for use, the authorization server MUST construct an error response as defined in **The OAuth 2.0 Authorization Protocol** [I-D.ietf.oauth-v2]. 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"
}
```

TOC

3.2. Client Authentication Processing

If the client JWT is not valid, or its subject confirmation requirements cannot be met, the authorization server MUST construct an error response as defined in **The OAuth 2.0 Authorization Protocol** [I-D.ietf.oauth-v2]. 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.

TOC

4. Authorization Grant Example

Though non-normative, the following examples illustrate what a conforming JWT and access token request would look like.

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

```
{"iss":"https://jwt-idp.example.com",
 "prn":"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 ECDSA P-256 SHA-256 algorithm.

```
{"alg":"ES256"}
```

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 long lines broken 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=eyJhbGciOiJFUzI1NiJ9.
eyJpc3Mi[...omitted for brevity...].
J9l-ZhwP_2n[...omitted for brevity...]
```

---

5. Security Considerations

TOC

No additional security considerations apply beyond those described within **The OAuth 2.0 Authorization Protocol** [I-D.ietf.oauth-v2], **OAuth 2.0 Assertion Profile** [I-D.ietf.oauth-assertions], and **JSON Web Token (JWT)** [JWT].

---

6. IANA Considerations

TOC

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

TOC

This is a request to IANA to please register the value grant-type:jwt-bearer in the registry urn:ietf:params:oauth established in **An IETF URN Sub-Namespace for OAuth** [I-D.ietf.oauth-urn-sub-ns].

- URN: urn:ietf:params:oauth:grant-type:jwt-bearer
- Common Name: JWT Bearer Token Grant Type Profile for OAuth 2.0
- Change controller: IETF
- Description: [[this document]]

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

TOC

This is a request to IANA to please register the value client-assertion-type:jwt-bearer in the registry urn:ietf:params:oauth established in **An IETF URN Sub-Namespace for OAuth** [I-D.ietf.oauth-urn-sub-ns].

- URN: urn:ietf:params:oauth:client-assertion-type:jwt-bearer
- Common Name: JWT Bearer Token Profile for OAuth 2.0 Client Authentication
- Change controller: IETF

- Description: [[this document]]

7. References

TOC

7.1. Normative References

TOC

[I-D.ietf.oauth-assertions]	Mortimore, C., Ed., Campbell, B., Jones, M., and Y. Goland, "OAuth 2.0 Assertion Profile," ID draft-ietf-oauth-assertions-01 (work in progress), October 2011 (**TXT**, **HTML**).
[I-D.ietf.oauth-urn-sub-ns]	Campbell, B., Ed. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth," ID draft-ietf-oauth-urn-sub-ns-00 (work in progress), Aug 2011 (**TXT**, **HTML**).
[I-D.ietf.oauth-v2]	Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Protocol," ID draft-ietf-oauth-v2-22 (work in progress), September 2011 (**TXT**, **HTML**).
[JWT]	**Jones, M.**, **Balfanz, D.**, **Bradley, J.**, **Goland, Y.**, **Panzer, J.**, **Sakimura, N.**, and **P. Tarjan**, "**JSON Web Token (JWT)**," December 2011.
[RFC2119]	**Bradner, S.**, "**Key words for use in RFCs to Indicate Requirement Levels**," BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).
[RFC4627]	Crockford, D., "**The application/json Media Type for JavaScript Object Notation (JSON)**," RFC 4627, July 2006 (**TXT**).

7.2. Informative References

TOC

[I-D.ietf-oauth-saml2-bearer]	Mortimore, C., "**SAML 2.0 Bearer Assertion Profiles for OAuth 2.0**," draft-ietf-oauth-saml2-bearer-09 (work in progress), October 2011 (**TXT**).

Appendix A. Acknowledgements

TOC

This profile was derived from **SAML 2.0 Bearer Assertion Profiles for OAuth 2.0** [I-D.ietf-oauth-saml2-bearer] by Brian Campbell and Chuck Mortimore.

Appendix B. Document History

TOC

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

-03

- Added the `jti` (JWT ID) claim to enable replay protection.
- Respect line length restrictions in examples.

-02

- Removed remaining vestiges of normative text talking about SAML that remained from the SAML Profile draft.
- 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]").

-01

- Merged in changes from draft-ietf-oauth-saml2-bearer-09. In particular, this draft now uses draft-ietf-oauth-assertions, rather than being standalone. It also now defines how to use JWT bearer tokens both for Authorization Grants and for Client Authentication.

-00

- Initial draft.

Authors' Addresses

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

Brian Campbell
Ping Identity Corp.
Email: brian.d.campbell@gmail.com

Chuck Mortimore
Salesforce.com
Email: cmortimore@salesforce.com