OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2015

B. Campbell
Ping Identity
C. Mortimore
Salesforce
M. Jones
Y. Goland
Microsoft

October 21, 2014

Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants
draft-ietf-oauth-assertions-18

Abstract

This specification provides a framework for the use of assertions with OAuth 2.0 in the form of a new client authentication mechanism and a new authorization grant type. Mechanisms are specified for transporting assertions during interactions with a token endpoint, as well as general processing rules.

The intent of this specification is to provide a common framework for OAuth 2.0 to interwork with other identity systems using assertions, and to provide alternative client authentication mechanisms.

Note that this specification only defines abstract message flows and processing rules. In order to be implementable, companion specifications are necessary to provide the corresponding concrete instantiations.

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 April 24, 2015.

Table of Contents

1. Introduction

An assertion is a package of information that facilitates the sharing of identity and security information across security domains. Section 3 provides a more detailed description of the concept of an assertion for the purpose of this specification.

OAuth 2.0 [RFC6749] is an authorization framework that enables a third-party application to obtain limited access to a protected HTTP resource. In OAuth, those third-party applications are called clients; they access protected resources by presenting an access token to the HTTP resource. Access tokens are issued to clients by an authorization server with the (sometimes implicit) approval of the resource owner. These access tokens are typically obtained by exchanging an authorization grant, which represents the authorization granted by the resource owner (or by a privileged administrator). Several authorization grant types are defined to support a wide range of client types and user experiences. OAuth also provides an extensibility mechanism for defining additional grant types, which can serve as a bridge between OAuth and other protocol frameworks.

This specification provides a general framework for the use of assertions as authorization grants with OAuth 2.0. It also provides a framework for assertions to be used for client authentication. It provides generic mechanisms for transporting assertions during interactions with an authorization server's token endpoint, as well as general rules for the content and processing of those assertions. The intent is to provide an alternative client authentication mechanism (one that doesn't send client secrets), as well as to facilitate the use of OAuth 2.0 in client-server integration scenarios, where the end-user may not be present.

This specification only defines abstract message flows and processing rules. In order to be implementable, companion specifications are necessary to provide the corresponding concrete instantiations. For instance, SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants [I-D.ietf-oauth-saml2-bearer] defines a concrete instantiation for SAML 2.0 assertions and JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants [I-D.ietf-oauth-jwt-bearer] defines a concrete instantiation for JWTs.

Note: The use of assertions for client authentication is orthogonal to and separable from using assertions as an authorization grant. They can be used either in combination or separately. Client assertion authentication 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. Assertion authorization grants may be used with or without client authentication or identification. Whether or not client authentication is needed in conjunction with an assertion authorization grant, as well as the supported types of client authentication, are policy decisions at the discretion of the authorization server.

2. 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 [RFC2119] .

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes must not be used as part of the value.

3. Framework

An assertion is a package of information that allows identity and security information to be shared across security domains. An assertion typically contains information about a subject or principal, information about the party that issued the assertion and when was it issued, as well as the conditions under which the assertion is to be considered valid, such as when and where it can be used.

The entity that creates and signs or integrity protects the assertion is typically known as the "Issuer" and the entity that consumes the assertion and relies on its information is typically known as the "Relying Party". In the context of this document, the authorization server acts as a relying party.

Assertions used in the protocol exchanges defined by this specification MUST always be integrity protected using a digital signature or Message Authentication Code applied by the issuer, which authenticates the issuer and ensures integrity of the assertion content. In many cases, the assertion is issued by a third party and it must be protected against tampering by the client that presents it. An assertion MAY additionally be encrypted, preventing unauthorized parties (such as the client) from inspecting the content.

Although this document does not define the processes by which the client obtains the assertion (prior to sending it to the authorization server), there are two common patterns described below.

In the first pattern, depicted in Figure 1, the client obtains an assertion from a third party entity capable of issuing, renewing, transforming, and validating security tokens. Typically such an entity is known as a "Security Token Service" (STS) or just "Token Service" and a trust relationship (usually manifested in the exchange of some kind of key material) exists between the token service and the relying party. The token service is the assertion issuer; its role is to fulfill requests from clients, which present various credentials, and mint assertions as requested, fill them with appropriate information, and integrity protect them with a signature or message authentication code. WS-Trust [OASIS.WS-Trust] is one available standard for requesting security tokens (assertions).

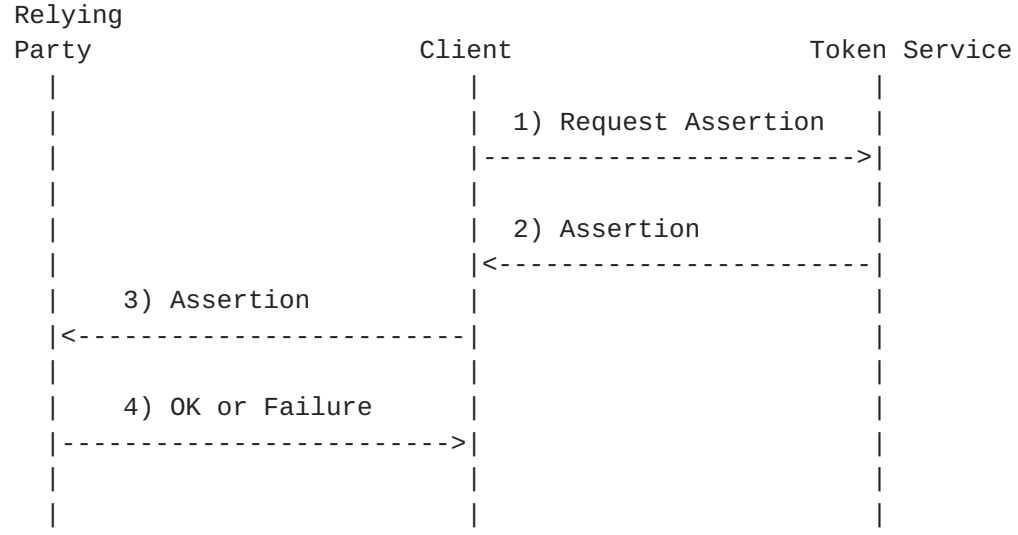

Figure 1: Third Party Created Assertion

In the second pattern, depicted in Figure 2, the client creates assertions locally. To apply the signatures or message authentication codes to assertions, it has to obtain key material: either symmetric keys or asymmetric key pairs. The mechanisms for obtaining this key material are beyond the scope of this specification.

Although assertions are usually used to convey identity and security information, self-issued assertions can also serve a different purpose. They can be used to demonstrate knowledge of some secret, such as a client secret, without actually communicating the secret directly in the transaction. In that case, additional information included in the assertion by the client itself will be of limited value to the relying party and, for this reason, only a bare minimum of information is typically included in such an assertion, such as information about issuing and usage conditions.

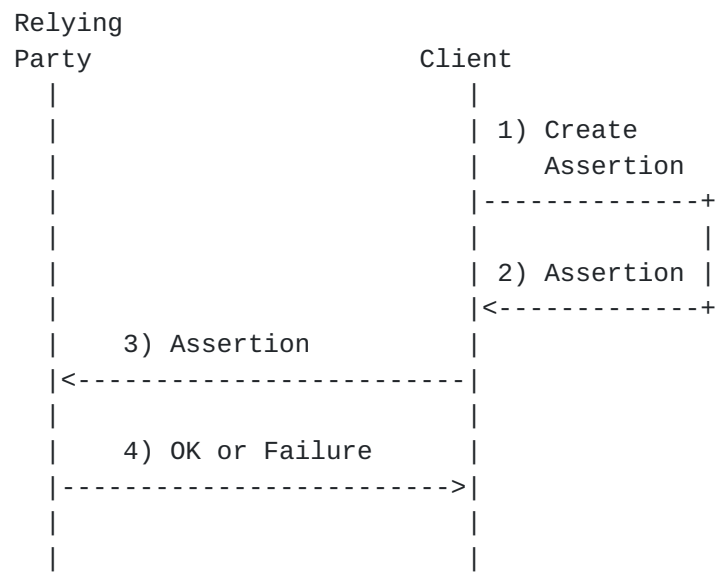

Figure 2: Self-Issued Assertion

Deployments need to determine the appropriate variant to use based on the required level of security, the trust relationship between the entities, and other factors.

From the perspective of what must be done by the entity presenting the assertion, there are two general types of assertions:

1. Bearer Assertions: Any entity in possession of a bearer assertion (the bearer) can use it to get access to the associated resources (without demonstrating possession of a cryptographic key). To prevent misuse, bearer assertions need to be protected from disclosure in storage and in transport. Secure communication channels are required between all entities to avoid leaking the assertion to unauthorized parties.

2. Holder-of-Key Assertions: To access the associated resources, the entity presenting the assertion must demonstrate possession of additional cryptographic material. The token service thereby binds a key identifier to the assertion and the client has to demonstrate to the relying party that it knows the key corresponding to that identifier when presenting the assertion.

The protocol parameters and processing rules defined in this document are intended to support a client presenting a bearer assertion to an authorization server. They are not directly suitable for use with holder-of-key assertions. While they could be used as a baseline for a holder-of-key assertion system, there would be a need for additional mechanisms (to support proof-of-possession of the secret

key), and possibly changes to the security model (e.g., to relax the requirement for an Audience).

4. Transporting Assertions

This section defines HTTP parameters for transporting assertions during interactions with a token endpoint of an OAuth authorization server. Because requests to the token endpoint result in the transmission of clear-text credentials (in both the HTTP request and response), all requests to the token endpoint MUST use TLS, as mandated in Section 3.2 of OAuth 2.0 [RFC6749].

4.1. Using Assertions as Authorization Grants

This section defines the use of assertions as authorization grants, based on the definition provided in Section 4.5 of OAuth 2.0 [RFC6749]. When using assertions as authorization grants, the client includes the assertion and related information using the following HTTP request parameters:

grant_type
: REQUIRED. The format of the assertion as defined by the authorization server. The value will be an absolute URI.

assertion
: REQUIRED. The assertion being used as an authorization grant. Specific serialization of the assertion is defined by profile documents.

scope
: OPTIONAL. The requested scope as described in Section 3.3 of OAuth 2.0 [RFC6749]. When exchanging assertions for access tokens, the authorization for the token has been previously granted through some out-of-band mechanism. As such, the requested scope MUST be equal or lesser than the scope originally granted to the authorized accessor. The Authorization Server MUST limit the scope of the issued access token to be equal or lesser than the scope originally granted to the authorized accessor.

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 assertion being used as an authorization grant (with extra line breaks for display purposes only):

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

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&
assertion=PHNhbWxwOl...[omitted for brevity]...ZT4
```

An assertion used in this context is generally a short lived representation of the authorization grant and authorization servers SHOULD NOT issue access tokens with a lifetime that exceeds the validity period of the assertion by a significant period. In practice, that will usually mean that refresh tokens are not issued in response to assertion grant requests and access tokens will be issued with a reasonably short lifetime. Clients can refresh an expired access token by requesting a new one using the same assertion, if it is still valid, or with a new assertion.

An IETF URN for use as the "grant_type" value can be requested using the template in [RFC6755]. A URN of the form urn:ietf:params:oauth:grant-type:* is suggested.

4.1.1. Error Responses

If an assertion is not valid or has expired, 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 assertion 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"
}
```

4.2. Using Assertions for Client Authentication

The following section defines the use of assertions as client credentials as an extension of Section 2.3 of OAuth 2.0 [RFC6749]. When using assertions as client credentials, the client includes the assertion and related information using the following HTTP request parameters:

client_assertion_type
REQUIRED. The format of the assertion as defined by the authorization server. The value will be an absolute URI.

client_assertion
REQUIRED. The assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents.

client_id
OPTIONAL. The client identifier as described in Section 2.2 of OAuth 2.0 [RFC6749]. The "client_id" is unnecessary for client assertion authentication because the client is identified by the subject of the assertion. If present, the value of the "client_id" parameter MUST identify the same client as is identified by the client assertion.

The following example demonstrates a client authenticating using an assertion during an Access Token Request, as defined in Section 4.1.3 of OAuth 2.0 [RFC6749] (with extra line breaks for display purposes only):

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

grant_type=authorization_code&
code=i1WsRn1uB1&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth
%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhbW...[omitted for brevity]...ZT
```

Token endpoints can differentiate between assertion based credentials and other client credential types by looking for the presence of the "client_assertion" and "client_assertion_type" parameters, which will only be present when using assertions for client authentication.

An IETF URN for use as the "client_assertion_type" value may be requested using the template in [RFC6755]. A URN of the form urn:ietf:params:oauth:client-assertion-type:* is suggested.

4.2.1. Error Responses

If an assertion is invalid for any reason or if more than one client authentication mechanism is used, 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 client assertion 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_client"
  "error_description":"assertion has expired"
}
```

5. Assertion Content and Processing

This section provides a general content and processing model for the use of assertions in OAuth 2.0 [RFC6749].

5.1. Assertion Metamodel

The following are entities and metadata involved in the issuance, exchange, and processing of assertions in OAuth 2.0. These are general terms, abstract from any particular assertion format. Mappings of these terms into specific representations are provided by profiles of this specification.

Issuer
: A unique identifier for the entity that issued the assertion. Generally this is the entity that holds the key material used to sign or integrity protect the assertion. Examples of issuers are OAuth clients (when assertions are self-issued) and third party security token services. If the assertion is self-issued, the Issuer value is the client identifier. If the assertion was issued by a Security Token Service (STS), the Issuer should identify the STS in a manner recognized by the Authorization Server. 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].

Subject
: A unique identifier for the principal that is the subject of the assertion.

  * When using assertions for client authentication, the Subject identifies the client to the authorization server using the value of the "client_id" of the OAuth client.

* When using assertions as an authorization grant, the Subject identifies an authorized accessor for which the access token is being requested (typically the resource owner, or an authorized delegate).

Audience
: A value that identifies the party or parties intended to process the assertion. The URL of the Token Endpoint, as defined in Section 3.2 of OAuth 2.0 [RFC6749], can be used to indicate that the authorization server as a valid intended audience of the assertion. 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].

Issued At
: The time at which the assertion was issued. While the serialization may differ by assertion format, it is REQUIRED that the time be expressed in UTC with no time zone component.

Expires At
: The time at which the assertion expires. While the serialization may differ by assertion format, it is REQUIRED that the time be expressed in UTC with no time zone component.

Assertion ID
: A nonce or unique identifier for the assertion. The Assertion ID may be used by implementations requiring message de-duplication for one-time use assertions. Any entity that assigns an identifier MUST ensure that there is negligible probability that that entity or any other entity will accidentally assign the same identifier to a different data object.

5.2. General Assertion Format and Processing Rules

The following are general format and processing rules for the use of assertions in OAuth:

- The assertion MUST contain an Issuer. The Issuer identifies the entity that issued the assertion as recognized by the Authorization Server. If an assertion is self-issued, the Issuer MUST be the value of the client's "client_id".

- The assertion MUST contain a Subject. 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. When the client is acting

on behalf of itself, the Subject MUST be the value of the client's "client_id".

- The assertion MUST contain an Audience that identifies the Authorization Server as the intended audience. The Authorization Server MUST reject any assertion that does not contain the its own identity as the intended audience.

- The assertion MUST contain an Expires At entity that limits the time window during which the assertion can be used. The authorization server MUST reject assertions that have expired (subject to allowable clock skew between systems). Note that the authorization server may reject assertions with an Expires At attribute value that is unreasonably far in the future.

- The assertion MAY contain an Issued At entity containing the UTC time at which the assertion was issued.

- The Authorization Server MUST reject assertions with an invalid signature or Message Authentication Code. The algorithm used to validate the signature or message authentication code and the mechanism for designating the secret used to generate the signature or message authentication code over the assertion are beyond the scope of this specification.

6. Common Scenarios

The following provides additional guidance, beyond the format and processing rules defined in Section 4 and Section 5, on assertion use for a number of common use cases.

6.1. Client Authentication

A client uses an assertion to authenticate to the authorization server's token endpoint by using the "client_assertion_type" and "client_assertion" parameters as defined in Section 4.2. The Subject of the assertion identifies the client. If the assertion is self-issued by the client, the Issuer of the assertion also identifies the client.

The example in Section 4.2 shows a client authenticating using an assertion during an Access Token Request.

6.2. Client Acting on Behalf of Itself

When a client is accessing resources on behalf of itself, it does so in a manner analogous to the Client Credentials Grant defined in Section 4.4 of OAuth 2.0 [RFC6749]. This is a special case that

combines both the authentication and authorization grant usage patterns. In this case, the interactions with the authorization server should be treated as using an assertion for Client Authentication according to Section 4.2, while using the grant_type parameter with the value "client_credentials" to indicate that the client is requesting an access token using only its client credentials.

The following example demonstrates an assertion being used for a Client Credentials Access Token Request, as defined in Section 4.4.2 of OAuth 2.0 [RFC6749] (with extra line breaks for display purposes only):

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

grant_type=client_credentials&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth
%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhbW...[omitted for brevity]...ZT
```

6.3. Client Acting on Behalf of a User

When a client is accessing resources on behalf of a user, it does so by using the "grant_type" and "assertion" parameters as defined in Section 4.1. The Subject identifies an authorized accessor for which the access token is being requested (typically the resource owner, or an authorized delegate).

The example in Section 4.1 shows a client making an Access Token Request using an assertion as an Authorization Grant.

6.3.1. Client Acting on Behalf of an Anonymous User

When a client is accessing resources on behalf of an anonymous user, a mutually agreed upon Subject identifier indicating anonymity is used. The Subject value might be an opaque persistent or transient pseudonymous identifier for the user or be an agreed upon static value indicating an anonymous user (e.g., "anonymous"). The authorization may be based upon additional criteria, such as additional attributes or claims provided in the assertion. For example, a client might present an assertion from a trusted issuer asserting that the bearer is over 18 via an included claim. In this case, no additional information about the user's identity is included, yet all the data needed to issue an access token is present.

More information about anonymity, pseudonymity, and privacy considerations in general can be found in [RFC6973].

7. Interoperability Considerations

This specification defines a framework for using assertions with OAuth 2.0. However, as an abstract framework in which the data formats used for representing many values are not defined, on its own, this specification is not sufficient to produce interoperable implementations.

Two other specifications that profile this framework for specific assertion have been developed: one [I-D.ietf-oauth-saml2-bearer] uses SAML 2.0-based assertions and the other [I-D.ietf-oauth-jwt-bearer] uses JSON Web Tokens (JWTs). These two instantiations of this framework specify additional details about the assertion encoding and processing rules for using those kinds of assertions with OAuth 2.0.

However, even when profiled for specific assertion types, agreements between system entities regarding identifiers, keys, and endpoints are required in order to achieve interoperable deployments. Specific items that require agreement are as follows: values for the issuer and audience identifiers, supported assertion and client authentication types, the location of the token endpoint, the key used to apply and verify the digital signature or Message Authentication Code over the assertion, one-time use restrictions on assertions, maximum assertion lifetime allowed, and the specific subject and attribute requirements of the assertion. The exchange of such information is explicitly out of scope for this specification. Deployments for particular trust frameworks, circles of trust, or other uses cases will need to agree among the participants on the kinds of values to be used for some abstract fields defined by this specification. In some cases, additional profiles may be created that constrain or prescribe these values or specify how they are to be exchanged. The OAuth 2.0 Dynamic Client Registration Core Protocol [I-D.ietf-oauth-dyn-reg] is one such profile that enables OAuth Clients to register metadata about themselves at an Authorization Server.

8. Security Considerations

This section discusses security considerations that apply when using assertions with OAuth 2.0 as described in this document. As discussed in Section 3, there are two different ways to obtain assertions: either as self-issued or obtained from a third party token service. While the actual interactions for obtaining an assertion are outside the scope of this document, the details are important from a security perspective. Section 3 discusses the high

level architectural aspects. Many of the security considerations discussed in this section are applicable to both the OAuth exchange as well as the client obtaining the assertion.

The remainder of this section focuses on the exchanges that concern presenting an assertion for client authentication and for the authorization grant.

8.1. Forged Assertion

Threat:
An adversary could forge or alter an assertion in order to obtain an access token (in case of the authorization grant) or to impersonate a client (in case of the client authentication mechanism).

Countermeasures:
To avoid this kind of attack, the entities must assure that proper mechanisms for protecting the integrity of the assertion are employed. This includes the issuer digitally signing the assertion or computing a keyed message digest over the assertion.

8.2. Stolen Assertion

Threat:
An adversary may be able obtain an assertion (e.g., by eavesdropping) and then reuse it (replay it) at a later point in time.

Countermeasures:
The primary mitigation for this threat is the use of secure communication channels with server authentication for all network exchanges.

An assertion may also contain several elements to prevent replay attacks. There is, however, a clear tradeoff between reusing an assertion for multiple exchanges and obtaining and creating new fresh assertions.

Authorization Servers and Resource Servers may use a combination of the Assertion ID and Issued At/Expires At attributes for replay protection. Previously processed assertions may be rejected based on the Assertion ID. The addition of the validity window relieves the authorization server from maintaining an infinite state table of processed Assertion IDs.

8.3. Unauthorized Disclosure of Personal Information

Threat:
The ability for other entities to obtain information about an individual, such as authentication information, role in an organization, or other authorization relevant information, raises privacy concerns.

Countermeasures:
To address the threats, two cases need to be differentiated:

First, a third party that did not participate in any of the exchange is prevented from eavesdropping on the content of the assertion by employing confidentiality protection of the exchange using TLS. This ensures that an eavesdropper on the wire is unable to obtain information. However, this does not prevent legitimate protocol entities from obtaining information that they are not allowed to possess from assertions. Some assertion formats allow for the assertion to be encrypted, preventing unauthorized parties from inspecting the content.

Second, an Authorization Server may obtain an assertion that was created by a third party token service and that token service may have placed attributes into the assertion. To mitigate potential privacy problems, prior consent for the release of such attribute information from the resource owner should be obtained. OAuth itself does not directly provide such capabilities, but this consent approval may be obtained using other identity management protocols, user consent interactions, or in an out-of-band fashion.

For the cases where a third party token service creates assertions to be used for client authentication, privacy concerns are typically lower, since many of these clients are Web servers rather than individual devices operated by humans. If the assertions are used for client authentication of devices or software that can be closely linked to end users, then privacy protection safeguards need to be taken into consideration.

Further guidance on privacy friendly protocol design can be found in [RFC6973].

8.4. Privacy Considerations

An assertion may contain privacy-sensitive information and, to prevent disclosure of such information to unintended parties, should only be transmitted over encrypted channels, such as TLS. In cases

where it is desirable to prevent disclosure of certain information the client, the assertion, or portions of it, should be be encrypted to the authorization server.

Deployments should determine the minimum amount of information necessary to complete the exchange and include only such information in the assertion. In some cases, the subject identifier can be a value representing an anonymous or pseudonymous user, as described in Section 6.3.1.

9. IANA Considerations

This is a request to add three values, as listed in the sub-sections below, to the "OAuth Parameters" registry established by RFC 6749 [RFC6749].

9.1. assertion Parameter Registration

- Parameter name: assertion
- Parameter usage location: token request
- Change controller: IESG
- Specification document(s): [[this document]]

9.2. client_assertion Parameter Registration

- Parameter name: client_assertion
- Parameter usage location: token request
- Change controller: IESG
- Specification document(s): [[this document]]

9.3. client_assertion_type Parameter Registration

- Parameter name: client_assertion_type
- Parameter usage location: token request
- Change controller: IESG
- Specification document(s): [[this document]]

10. References

10.1. Normative References

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

10.2. Informative References

[I-D.ietf-oauth-dyn-reg]
Richer, J., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", draft-ietf-oauth-dyn-reg-20 (work in progress), August 2014.

[I-D.ietf-oauth-jwt-bearer]
Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", draft-ietf-oauth-jwt-bearer (work in progress), October 2014.

[I-D.ietf-oauth-saml2-bearer]
Campbell, B., Mortimore, C., and M. Jones, "SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants", draft-ietf-oauth-saml2-bearer (work in progress), October 2014.

[OASIS.WS-Trust]
Nadalin, A., Ed., Goodner, M., Ed., Gudgin, M., Ed., Barbir, A., Ed., and H. Granqvist, Ed., "WS-Trust", Feb 2009.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, October 2012.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

Appendix A. Acknowledgements

The authors wish to thank the following people that have influenced or contributed this specification: Paul Madsen, Eric Sachs, Jian Cai, Tony Nadalin, Hannes Tschofenig, the authors of the OAuth WRAP specification, and the members of the OAuth working group.

Appendix B. Document History

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

draft-ietf-oauth-assertions-18

- Changes/suggestions from IESG reviews.

draft-ietf-oauth-assertions-17

- Added Privacy Considerations section per AD review discussion http://www.ietf.org/mail-archive/web/oauth/current/msg13148.html and http://www.ietf.org/mail-archive/web/oauth/current/msg13144.html

draft-ietf-oauth-assertions-16

- Clarified some text around the treatment of subject based on the rough rough consensus from the thread staring at http://www.ietf.org/mail-archive/web/oauth/current/msg12630.html

draft-ietf-oauth-assertions-15

- Updated references.

- Improved formatting of hanging lists.

draft-ietf-oauth-assertions-14

- Update reference: draft-iab-privacy-considerations is now RFC 6973

- Update reference: draft-ietf-oauth-dyn-reg from -13 to -15

draft-ietf-oauth-assertions-13

- Clean up language around subject per the subject part of http://www.ietf.org/mail-archive/web/oauth/current/msg12155.html

- Replace "Client Credentials flow" by "Client Credentials _Grant_" as suggested in http://www.ietf.org/mail-archive/web/oauth/current/msg12155.html

- For consistency with SAML and JWT per http://www.ietf.org/mail-archive/web/oauth/current/msg12251.html and http://www.ietf.org/mail-archive/web/oauth/current/msg12253.html Stated that "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."

- Added one-time use, maximum lifetime, and specific subject and attribute requirements to Interoperability Considerations.

draft-ietf-oauth-assertions-12

- Stated that issuer and audience values SHOULD be compared using the Simple String Comparison method defined in Section 6.2.1 of RFC 3986 unless otherwise specified by the application.

draft-ietf-oauth-assertions-11

- Addressed comments from IESG evaluation https://datatracker.ietf.org/doc/draft-ietf-oauth-assertions/ballot/.

- Reworded Interoperability Considerations to state what identifiers, keys, endpoints, etc. need to be exchanged/agreed upon.

- Added brief description of assertion to the into and included a reference to Section 3 (Framework) where it's described more.

- Changed such that a self-issued assertion must (was should) have the client id as the issuer.

- Changed "Specific Assertion Format and Processing Rules" to "Common Scenarios" and reworded to be more suggestive of common practices, rather than trying to be normative. Also removed lots of repetitive text in that section.

- Refined language around audience, subject, client identifiers, etc. to hopefully be clearer and less redundant.

- Changed title from "Assertion Framework for OAuth 2.0" to "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants" to be more explicit about the scope of the document per http://www.ietf.org/mail-archive/web/oauth/current/msg11063.html.

- Noted that authentication of the client per Section 3.2.1 of OAuth is optional for an access token request with an assertion as an

authorization grant and removed client_id from the associated example.

draft-ietf-oauth-assertions-10

- Changed term "Principal" to "Subject".
- Added Interoperability Considerations section.
- Applied Shawn Emery's comments from the security directorate review, including correcting urn:ietf:params:oauth:grant_type:* to urn:ietf:params:oauth:grant-type:*.

draft-ietf-oauth-assertions-09

- Allow audience values to not be URIs.
- Added informative references to draft-ietf-oauth-saml2-bearer and draft-ietf-oauth-jwt-bearer.
- Clarified that the statements about possible issuers are non-normative by using the language "Examples of issuers".

draft-ietf-oauth-assertions-08

- Update reference to RFC 6755 from draft-ietf-oauth-urn-sub-ns
- Tidy up IANA consideration section

draft-ietf-oauth-assertions-07

- Reference RFC 6749.
- Remove extraneous word per http://www.ietf.org/mail-archive/web/oauth/current/msg10029.html

draft-ietf-oauth-assertions-06

- Add more text to intro explaining that an assertion grant type can be used with or without client authentication/identification and that client assertion authentication is nothing more than an alternative way for a client to authenticate to the token endpoint

draft-ietf-oauth-assertions-05

- Non-normative editorial cleanups

draft-ietf-oauth-assertions-04

- Updated document to incorporate the review comments from the shepherd - thread and alternative draft at http://www.ietf.org/mail-archive/web/oauth/current/msg09437.html

- Added reference to draft-ietf-oauth-urn-sub-ns and include suggestions on urn:ietf:params:oauth:[grant-type|client-assertion-type]:* URNs

draft-ietf-oauth-assertions-03

- updated reference to draft-ietf-oauth-v2 from -25 to -26

draft-ietf-oauth-assertions-02

- Added text about limited lifetime ATs and RTs per http://www.ietf.org/mail-archive/web/oauth/current/msg08298.html.

- Changed the line breaks in some examples to avoid awkward rendering to text format. Also removed encoded '=' padding from a few examples because both known derivative specs, SAML and JWT, omit the padding char in serialization/encoding.

- Remove section 7 on error responses and move that (somewhat modified) content into subsections of section 4 broken up by authn/authz per http://www.ietf.org/mail-archive/web/oauth/current/msg08735.html.

- Rework the text about "MUST validate ... in order to establish a mapping between ..." per http://www.ietf.org/mail-archive/web/oauth/current/msg08872.html and http://www.ietf.org/mail-archive/web/oauth/current/msg08749.html.

- Change "The Principal MUST identify an authorized accessor. If the assertion is self-issued, the Principal SHOULD be the client_id" in 6.1 per http://www.ietf.org/mail-archive/web/oauth/current/msg08873.html.

- Update reference in 4.1 to point to 2.3 (rather than 3.2) of oauth-v2 (rather than self) http://www.ietf.org/mail-archive/web/oauth/current/msg08874.html.

- Move the "Section 3 of" out of the xref to hopefully fix the link in 4.1 and remove the client_id bullet from 4.2 per http://www.ietf.org/mail-archive/web/oauth/current/msg08875.html.

- Add ref to Section 3.3 of oauth-v2 for scope definition and remove some then redundant text per http://www.ietf.org/mail-archive/web/oauth/current/msg08890.html.

- Change "The following format and processing rules SHOULD be applied" to "The following format and processing rules apply" in sections 6.x to remove conflicting normative qualification of other normative statements per http://www.ietf.org/mail-archive/web/oauth/current/msg08892.html.

- Add text the client_id must id the client to 4.1 and remove similar text from other places per http://www.ietf.org/mail-archive/web/oauth/current/msg08893.html.

- Remove the MUST from the text prior to the HTTP parameter definitions per http://www.ietf.org/mail-archive/web/oauth/current/msg08920.html.

- Updated examples to use grant_type and client_assertion_type values from the OAuth SAML Assertion Profiles spec.

Authors' Addresses

Brian Campbell
Ping Identity

Email: brian.d.campbell@gmail.com

Chuck Mortimore
Salesforce.com

Email: cmortimore@salesforce.com

Michael B. Jones
Microsoft

Email: mbj@microsoft.com

Yaron Y. Goland
Microsoft

Email: yarong@microsoft.com