OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019
D. Hardt
Amazon
October 19, 2018

Reciprocal OAuth
draft-ietf-oauth-reciprocal-01

Abstract

There are times when a user has a pair of protected resources that would like to request access to each other. While OAuth flows typically enable the user to grant a client access to a protected resource, granting the inverse access requires an additional flow. Reciprocal OAuth enables a more seamless experience for the user to grant access to a pair of protected resources.

Status of This Memo

Copyright Notice

1. Introduction

In the usual three legged, authorization code grant, the OAuth flow enables a resource owner (user) to enable a client (party A) to be granted authorization to access a protected resource (party B). If party A also has a protected resource that the user would like to let party B access, then a second complete OAuth flow, but in the reverse direction, must be performed. In practice, this is a complicated user experience as the user is at Party A, but the OAuth flow needs to start from Party B. This requires the second flow to send the user back to party B, which then sends the user to Party A as the first step in the flow. At the end, the user is at Party B, even though the original flow started at Party A.

Reciprocal OAuth simplifies the user experience by eliminating the redirections in the second OAuth flow. After the intial OAuth flow, party A obtains consent from the user to grant party B access to a protected resource at party A, and then passes an authorization code to party B using the access token party A obtained from party B to provide party B the context of the user. Party B then exchanges the authorization code for an access token per the usual OAuth flow.

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Reciprocal Scope Request

When party B is providing an access token response per [RFC6749] 4.1.4, 4.2.1, 4.3.3 or 4.4.3, party B MAY include an additional query component in the redirection URI to indicate the scope requested in the reciprocal grant.

reciprocal OPTIONAL. The scope of party B's reciprocal access request per [RFC6749] 3.3.

If party B does not provide a reciprocal parameter in the access token response, the reciprocal scope will be a value previously preconfigured by party A and party B.

If an authorization code grant access token response per [RFC6749] 4.1.4, an example successful response:

```
     HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: no-store
     Pragma: no-cache

     {
       "access_token":"2YotnFZFEjr1zCsicMWpAA",
       "token_type":"example",
       "expires_in":3600,
       "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
       "reciprocal":"example_scope",
       "example_parameter":"example_value"
     }
```

If an authorization code grant access token response per [RFC6749] 4.2.2, an example successful response (with extra line breaks for display purposes only):

```
     HTTP/1.1 302 Found
     Location: http://example.com/cb#
         access_token=2YotnFZFEjr1zCsicMWpAA&
         state=xyz&token_type=example&
         expires_in=3600&
         reciprocal="example_scope"
```

3. Reciprocal Authorization Flow

The reciprocal authorization flow starts after the client (party A) has obtained an access token from the authorization server (party B) per [RFC6749] 4.1 Authorization Code Grant.

3.1. User Consent

Party A obtains consent from the user to grant Party B access to protected resources at party A. The consent represents the scopes party B had preconfigured at party A.

3.2. Reciprocal Authorization Code

Party A generates an authorization code representing the access granted to party B by the user. Party A then makes a request to party B's token endpoint authenticating per [RFC6749] 2.3 and sending the following parameters using the "application/x-www-form-urlencoded" format per [RFC6749] Appendix B with a character encoding of UTF-8 in the HTTP request entity-body:

grant_type REQUIRED. Value MUST be set to "urn:ietf:params:oauth:grant-type:reciprocal".

code REQUIRED. The authorization code generated by party A.

client_id REQUIRED, party A'a client ID.

access_token REQUIRED, the access token obtained from Party B. Used to provide user context.

For example, the client makes the following HTTP request using TLS (with extra line breaks for display purposes only):

```
 POST /token HTTP/1.1
 Host: server.example.com
 Authorization: Basic ej4hsyfishwssjdusisdhkjsdksusdhjkjsdjk
 Content-Type: application/x-www-form-urlencoded

 grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-
type%3reciprocal&code=hasdyubasdjahsbdkjbasd&client_id=example.com&access_token=sadadojsadlkjas
```

Party B MUST verify the authentication provided by Party A per [RFC6749] 2.3

Party B MUST then verify the access token was granted to the client identified by the client_id.

Party B MUST respond with either an HTTP 200 (OK) response if the request is valid, or an HTTP 400 "Bad Request" if it is not.

Party B then plays the role of the client to make an access token request per [RFC6749] 4.1.3.

4. Authorization Update Flow

After the initial authorization, the user may add or remove scopes available to the client at the authorization server. For example, the user may grant additional scopes to the client using a voice interface, or revoke some scopes. The authorization server can update the client with the new authorization by sending a new authorization code per 3.2.

5. IANA Considerations

TBD.

6. Acknowledgements

TBD.

7. Normative References

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

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

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

Appendix A. Document History

A.1. draft-ietf-oauth-reciprical-00

- o Initial version.

A.2. draft-ietf-oauth-reciprical-01

- o changed reciprocal scope request to be in access token response rather than authorization request

Author's Address

Dick Hardt
Amazon

Email: dick.hardt@gmail.com