Reciprocal OAuth
draft-ietf-oauth-reciprocal-04

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

   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 https://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 February 2, 2020.

Copyright Notice

the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

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.

   For example, a user would like their voice assistant (party A) and
   music service (party B) to work together. The voice assistant wants
   to call the music service to play music, and the music service wants
   to call the voice assistant with music information to present to the
   user. The user starts the OAuth flow at the voice assistant, and is
   redirected to the music service. The music services obtains consent
   from the user and the redirects back to the voice assistant. At this
   point the voice assistant is able to obtain an access token for the
   music service. The voice assistant can the get consent from the user
   to authorize the music service to access the voice assistant, and
   then the voice assistant can create an authorization code and send it
   to the music service, which then exchanges the authorization code for
   an access token, all without further user interaction. Note that
   either the voice assistant or the music service can initiate the
   flow, so that either can prompt the user for the two parties to work
   together.

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 Protocol Flow

```
     Party A                                      Party B
     +--------------+                             +--------------+
     |              |--(A)- Authorization Request ->| Resource     |
     |              |                             |   Owner B    |
     |              |<-(B)-- Authorization Grant ---|              |
     |              |                             +--------------+
     |              |
     |   Client A   |                             +--------------+
     |              |                             |              |
     |              |--(C)-- Authorization Grant -->|              |
     |              |                             | Authorization|
     |              |<-(D)---- Access Token B ------|   Server B   |
     |              |         reciprocal Request  |              |
     +--------------+                             +--------------+
            |
     reciprocal Request
            V
     +--------------+                             +--------------+
     |   Resource   |                             | Authorization|
     |   Owner A    |--(E)--- reciprocal Grant ---->|   Server B   |
     |              |          Access Token B     |              |
     +--------------+                             +--------------+
                                                         |
                                                  reciprocal Grant
                                                         V
     +--------------+                             +--------------+
     |              |<-(F)--- reciprocal Grant -----|              |
     | Authorization|                             |   Client B   |
     |   Server A   |--(G)---- Access Token A ----->|              |
     +--------------+                             +--------------+
```
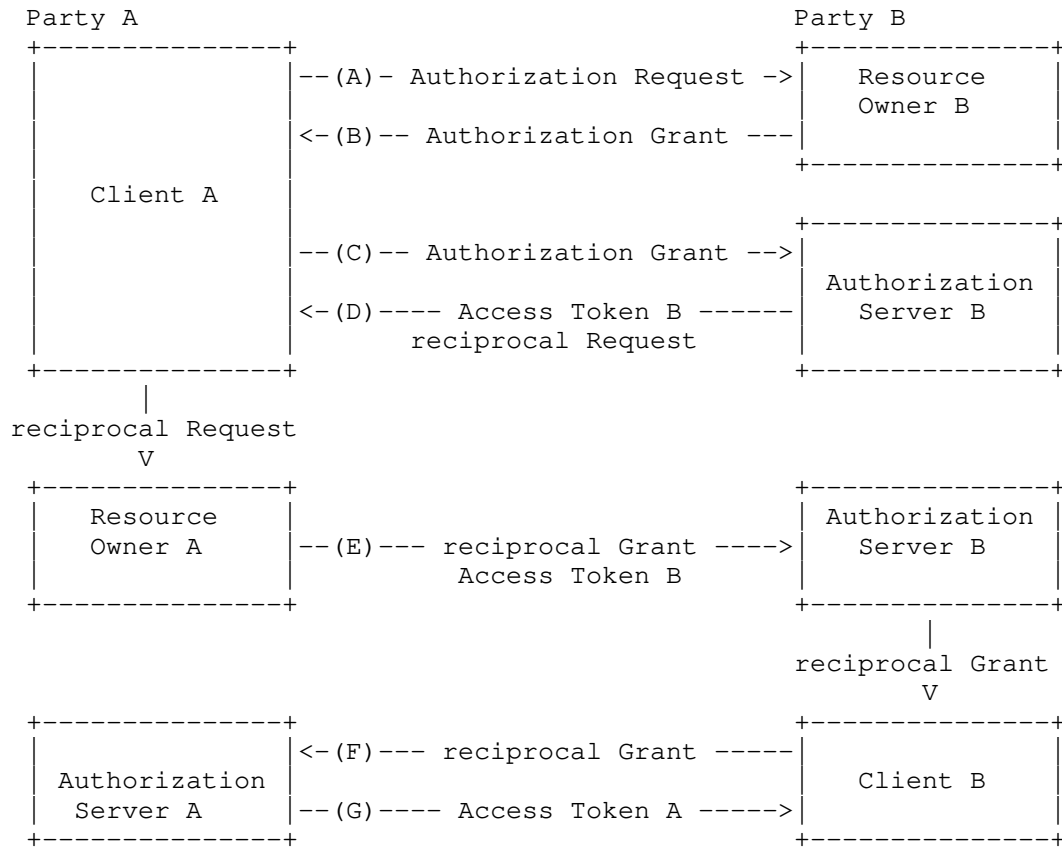
   Figure 1: Abstract reciprocal Protocol Flow

   The reciprocal authorization between party A and party B are
   abstractly represented in Figure 1 and includes the following steps:

   o (A - C) are the same as in [RFC6749] 1.2

   o (D) Party B optionally includes the reciprocal scope in the
      response. See Section 2.1 for details.

   o (E) Party A sends the reciprocal authorization grant to party B.
      See Section 2.2.2 for details.

   o (F) Party B requests an access token, mirroring step (B)

   o (G) Party A issues an access token, mirroring step (C)

Note that Resource Owner A and Resource Owner B are the respective
resource owner interaction systems controlled by the same owner.

2.1. 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 (with extra line breaks for
display purposes only):

```
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"
```

When party B is providing an authorization response per [RFC6749]
4.1.2, 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
authorization response, the reciprocal scope will be a value
previously preconfigured by party A and party B.

2.2. 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.

2.2.1. 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
requested by party B from party A per Section 2.1.

2.2.2. 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
   by Party B to identify which user authorization is being requested.

   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=sadadojsadlkjasdkljxxlkjdas
```

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.

3. 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 Section 2.2.2.

4. IANA Considerations

4.1. Registration of reciprocal

This section registers the value "reciprocal" in the IANA "OAuth
Parameters" registry established by "The OAuth 2.0 Authorization
Framework" [RFC6749].

o Parameter Name: reciprocal

o Parameter usage location: token response

o Change Controller: IESG

o Specification Document: Section Section 2.1 of this document

4.2. Sub-Namespace Registration of urn:ietf:params:oauth:grant-
      type:reciprocal

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

   o URN: urn:ietf:params:oauth:grant-type:reciprocal

   o Common Name: reciprocal grant for OAuth 2.0

   o Change Controller: IESG

   o Specification Document: Section Section 2.2.2 of this document

5. 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>.

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

Appendix A. Document History

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

   o Initial version.

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

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

A.3. draft-ietf-oauth-reciprocal-02

   o Added in diagram to clarify protocol flow

A.4. draft-ietf-oauth-reciprocal-03

   o fixed spelling of reciprocal

   o added example use case in introduction

   o resource owner is the same in Party A and Party B

A.5. draft-ietf-oauth-reciprocal-04

   o completed IANA section

Author's Address

   Dick Hardt

   Email: dick.hardt@gmail.com