Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 20, 2018

D. Hardt
Amazon

January 16, 2018

Reciprocal OAuth
draft-hardt-oauth-mutual-02

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 seemless 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 complete OAuth flow, but in the reverse direction, must be performed.

Reciprocal OAuth enables party A to obtain constent from the user to grant access to a protected resource at party A, and to short circuit the OAuth flow by passing an authorization code to party B using the acces token party A obtained from party B to provide party B the context of the user. This simplifies the user experience for each party to obtain acces tokens from the other.

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 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.

2.1. User Consent

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

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 to provide user context. [DH: security concern passing the access token in the body?]

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 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. IANA Considerations

TBD.

4. Acknowledgements

TBD.

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>.

Appendix A. Document History

A.1. draft-hardt-oauth-mutual-00

o Initial version.

A.2. draft-hardt-oauth-mutual-01

o renamed to Reciprocal OAuth

o clarified user consent in reciprocal flow

o changed authentication to be client authentication per [RFC6749] 2.3

A.3. draft-hardt-oauth-mutual-02

o changed grant type to URI

o added valid request response codes in 2.2

Author's Address

Dick Hardt
Amazon

Email: dick.hardt@gmail.com