OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2018

D. Hardt
Amazon
B. Campbell
Ping Identity
N. Sakimura
NRI

June 12, 2018

Distributed OAuth
draft-hardt-oauth-distributed-01

Abstract

The Distributed OAuth profile enables an OAuth client to discover what authorization server or servers may be used to obtain access tokens for a given resource, and what parameter values to provide in the access token request.

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 December 14, 2018.

1. Introduction

In [RFC6749], there is a single resource server and authorization server. In more complex and distributed systems, a clients may access many different resource servers, which have different authorization servers managing access. For example, a client may be accessing two different resources that provides similar functionality, but each is in a different geopolitical region, which requires authorization from authorization servers located in each geopolitical region.

A priori knowledge by the client of the relationships between resource servers and authorizations servers is not practical as the number of resource servers and authorization servers scales up. The client needs to discover on-demand which authorization server to request authorization for a given resource, and what parameters to pass. Being able to discover how to access a protected resource also enables more flexible software development as changes to the scopes, realms and authorization servers can happen dynamically with no change to client code.

1.1. Notational Conventions

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, [RFC2119].

1.2. Terminology

Issuer: the party issuing the access token, also known as the authorization server.

All other terms are as defined in [RFC6749] and [RFC6750]

1.3. Protocol Overview

Figure 1 shows an abstract flow of distributed OAuth.

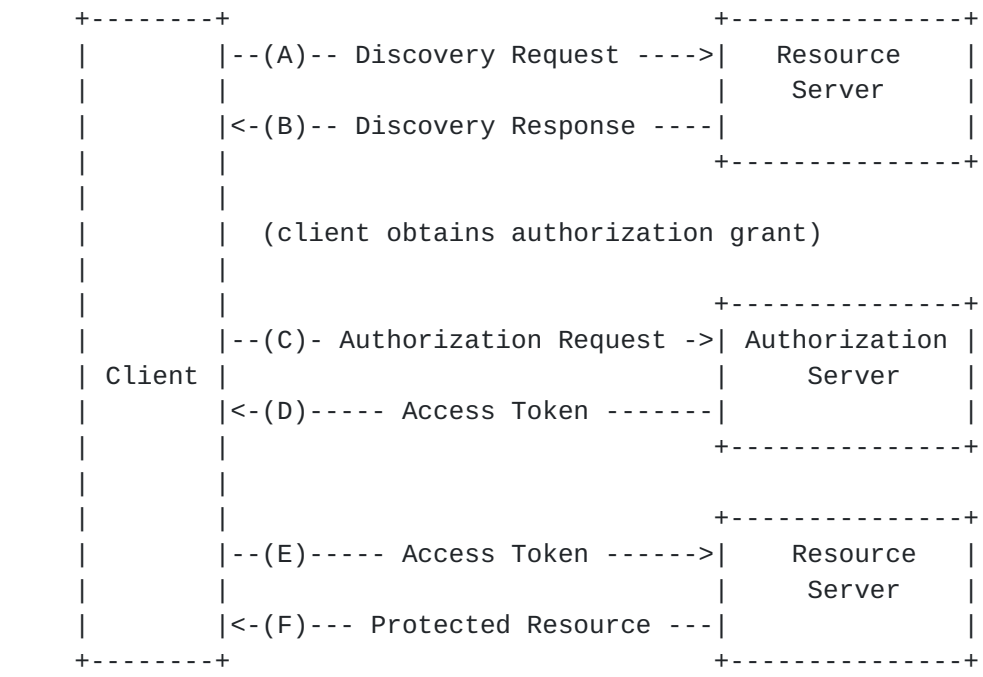

Figure 1: Abstract Protocol Flow

There are three steps where there are changes from the OAuth flow:

1) A discovery request (A) and discovery response (B) where the client discovers what is required to make an authenticated request. The client makes a request to the protected resource without supplying the Authorization header, or supplying an invalid access token. The resource server responds with a HTTP 401 response code and links of relation types "resource_uri" and the "oauth_server_metadata_uri". The client confirms the "host" value from the TLS connection is contained in the resource URI, and fetches each OAuth Server Metadata URI and per [OASM] discovers one or more authorization server end point URIs.

The client then obtains an authorization grant per one of the grant types in [RFC6749] section 4.

2) An authorization request (C) to an authorization server and includes the "resource_uri" link. The authorization servers provides an access token that is associated to the "resource_uri" value.

3) An authenticated request (E) to the resource server that confirms the "resource_uri" linked to the access token matches expected value.

2. Authorization Server Discovery

Figure 1, step (A)

To access a protected resource, the client needs to learn the authorization servers or issuers that can issue access tokens that are acceptable to the protected resource. There may be one or more issuers that can issue access tokens for the protected resource. To discover the issuers, the client attempts to make a call to the protected resource URI as defined in [RFC6750] section 2.1, except with an invalid access token or no HTTP "Authorization" request header field. The client notes the hostname of the protected resource that was confirmed by the TLS connection, and saves it as the "host" attribute.

Figure 1, step (B)

The resource server responds with the "WWW-Authenticate" HTTP header that includes the "error" attribute with a value of "invalid_token" and MAY also include the "scope" and "realm" attribute per [RFC6750] section 3, and a "Link" HTTP Header per [RFC8288] that MUST include one link of relation type "resource_uri" and one or more links of type "oauth_server_metadata_uri".

For example (with extra spaces and line breaks for display purposes only):

```
 HTTP/1.1 401 Unauthorized
 WWW-Authenticate: Bearer realm="example_realm",
                          scope="example_scope",
                          error="invalid_token"
 Link: <https://api.example.com/resource">; rel="resource_uri",
       <https://as.example.com/.well-known/oauth-authorization-server>;
rel="oauth_server_metadata_uri"
```

The client MUST confirm the host portion of the resource URI, as specified in the "resource_uri" link, contains the "host" attribute obtained from the TLS connection in step (A). The client MUST confirm the resource URI is contained in the protected resource URI where access was attempted. The client then retrieves one or more of the OAuth Server Metadata URIs to learn how to interact with the associated authorization server per [OASM] and create a list of one or more authorization server token endpoint URLs.

3. Authorization Grant

The client obtains an authorization grant per any of the mechanisms in [RFC6749] section 4.

4. Access Token Request

Figure 1, step (C)

The client makes an access token request to the authorization server token endpoint URL, or if more than URL is available, a randomly selected URL from the list. If the client is unable to connect to the URL, then the client MAY try to connect to another URL from the list.

The client SHOULD authenticate to the issuer using a proof of possession mechanism such as mutual TLS or a signed token containing the issuer as the audience.

Depending on the authorization grant mechanism used per [RFC6749] section 4, the client makes the access token request and MUST include "resource" as an additional parameter with the value of the resource URI. For example, if using the [RFC6749] section 4.4, Client Credentials Grant, the request would be (with extra spaces and line breaks for display purposes only):

```
POST /token HTTP/1.1
Host: issuer.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&scope=example_scope
&resource=https%3A%2F%2Fapi.example.com%2Fresource
```

Figure 1, step (D)

The authorization server MUST associate the resource URI with the issued access token in a way that can be accessed and verified by the protected resource. For JWT [RFC7519] formatted access tokens, the "aud" claim MUST be used to convey the resource URI. When Token Introspection [RFC7662] is used, the introspection response MUST containe the "aud" member with the resource URI as its value.

5. Accessing Protected Resource

Figure 1, step (E)

The client accesses the protected resource per [RFC6750] section 2.1. The Distributed OAuth Profile MUST only use the authorization request header field for passing the access token.

Figure 1, step (F)

The protected resource MUST verify the resource URI in or referenced by the access token is the protected resource's resource URI.

6. Security Considerations

Three new threats emerge when the client is dynamically discovering the authorization server and the request attributes: access token reuse, resource server impersonation, and malicious issuer.

6.1. Access Token Reuse

A malicious resource server impersonates the client and reuses the access token provided by the client to the malicious resource server with another resource server.

This is mitigated by constraining the access token to a specific audience, or to a specific client.

Audience restricting the access token is described in this document where the the resource URI is associated to the access token by inclusion or reference, so that only access tokens with the correct resource URI are accepted at a resource server.

Sender constraining the access token can be done through [MTLS], [OATB], or any other mechanism that the resource can use to associate the access token with the client.

6.2. Resource Server Impersonation

A malicious resource server tells a client to obtain an access token that can be used at a different resource server. When the client presents the access token, the malicious resource server uses the access token to access another resource server.

This is mitigated by the client obtaining the "host" value from the TLS certificate of the resource server, and the client verifying the "host" value is contained in the host portion of the resource URI, rather than the resource URI being any value declared by the resource server.

6.3. Malicious Issuer

A malicious resource server could redirect the client to a malicious issuer, or the issuer may be malicious. The malicious issuer may replay the client credentials with a valid issuer and obtain a valid access token for a protected resource.

This attack is mitigated by the client using a proof of possession authentication mechanism with the issuer such as [MTLS] or a signed token containing the issuer as the audience.

7. IANA Considerations

Pursuant to [RFC5988], the following link type registrations will be registered by mail to link-relations@ietf.org.

- Relation Name: oauth_server_metadata_uri
- Description: An OAuth 2.0 Server Metadata URI.
- Reference: This specification
- Relation Name: resource_uri
- Description: An OAuth 2.0 Resource Endpoint specified in [RFC6750] section 3.2.
- Reference: This specification

8. Acknowledgements

TBD.

9. Normative References

[MTLS] Jones, M., Campbell, B., Bradley, J., and W. Denniss, "OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens", June 2018, <https://datatracker.ietf.org/doc/draft-ietf-oauth-mtls/>.

[OASM] Jones, M., Campbell, B., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", June 2018, <https://datatracker.ietf.org/doc/draft-ietf-oauth-discovery/>.

[OATB] Campbell, B., Bradley, J., Sakimora, N., and T. Lodderstedt, "OAuth 2.0 Token Binding", June 2018, <https://datatracker.ietf.org/doc/draft-ietf-oauth-token-binding/>.

[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>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <https://www.rfc-editor.org/info/rfc5988>.

[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>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

Appendix A. Document History

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

- Initial version.

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

- resource identity expanded from just a hostname "host", to a URI that contains the hostname "resource URI"
- use oauth discovery document to obtain token endpoint rather than explicitly returning token endpoint
- use [RFC8288] to provide resource and discovery URIs
- allow any authorization grant type be used to obtain an authorization grant
- change attribute "host" to "resource"
- require linking resource URI to access token

- add client restriction to mitigate access token reuse

- added Nat and Brian as authors

Authors' Addresses

Dick Hardt
Amazon

Email: dick.hardt@gmail.com

Brian Campbell
Ping Identity

Email: brian.d.campbell@gmail.com

Nat Sakimura
NRI

Email: n-sakimura@nri.co.jp