Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

D. Hardt
Amazon
October 30, 2017

Distributed OAuth
draft-hardt-distributed-oauth-00

Abstract

The OAuth client credentials grant The Distributed OAuth profile enables an OAuth client using the client credentials grant to discover what authorization server to use for a given resource server, and what attribute values to provide in the access token request.

Status of This Memo

Copyright Notice

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 funcationility, but each is in a different geopolitical region, which requires authorization from authorization servers located in each geopolitical region.

A priori knownledge 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 attributes 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.

```
    +--------+                               +---------------+
    |        |--(A)-- Discovery Request ---->|   Resource    |
    |        |                               |    Server     |
    |        |<-(B)-- Discovery Response ----|               |
    |        |                               +---------------+
    |        |
    |        |                               +---------------+
    |        |--(C)- Authorization Request ->| Authorization |
    | Client |                               |     Server    |
    |        |<-(D)----- Access Token -------|               |
    |        |                               +---------------+
    |        |
    |        |                               +---------------+
    |        |--(E)----- Access Token ------>|    Resource   |
    |        |                               |     Server    |
    |        |<-(F)--- Protected Resource ---|               |
    +--------+                               +---------------+
```

Figure 1: Abstract Protocol Flow

There are three steps where there are changes from the OAuth client credential grant:

1) A discovery request (A) and discovery resopnse (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. The resource server responds with a HTTP 401 response code and the "iss" attribute in the "WWW-Authenticate" header. The client notes the "host" value from the TLS

2) An authorization request (C) to an authorization server listed in the "iss" attribute and includes the "host" attribute. The authorization servers provides an access token that includes the "host" value.

3) An authenticated request (E) to the resource server that confirms the "host" matches expected values.

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 makes an unauthenticated call to the protected resource, with no HTTP "Authorization" request header field

as defined in [RFC6750] section 2.1. 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 MAY incldue the "scope" and "realm" attribute per [RFC6750] section 3, and MUST include the "iss" attribute. The "iss" attribute has one or more space delimited URLs.

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

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example_realm",
                         iss="http://issuer.example.com/token",
                         scope="example_scope",
                         error="invalid_token"
```

3. Access Token Request

Figure 1, step (C)

The client makes an access token request to the issuer URL specified by the "iss" attribute, or if more than one value, a randomly selected URL from the list. If the client is unable to connect to the issuer, 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

The client makes the access token request per [RFC6749] section 4.4, Client Credentials Grant and MUST include the "host" attribute, for example (with extra 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"
&host=resource.example.com
&realm="example_realm"
```

Figure 1, step (D)

The authorization server returns an access token that MUST include the "host" attribute or a reference that includes the "host" attribute.

4. Accessing Protected Resource

Figure 1, step (E)

The client access the protected resource per [RFC6750] section 2.1. The distributed oauth profile SHALL only use the authorization request header field for passing the access token.

Figure 1, step (F)

The protected resource MUST verify the "host" attribute is the host of the resource server the protected resource is hosted.

5. New Threats

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.

5.1. 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 inclusion of the "host" property in the acces token, so that only the access token is only accepted at the intended host.

5.2. 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 acces 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 rather than being declared by the resource server.

5.3. 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 possion authentication mechanism with the issuer such as mutual TLS or a signed token containing the issuer as the audience.

6. IANA Considerations

TBD.

7. Acknowledgements

TBD.

8. 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-distributed-oauth-00

o Initial version.

Author's Address

Dick Hardt
Amazon

Email: dick.hardt@gmail.com