Workgroup: Web Authorization Protocol
Internet-Draft:
draft-bertocci-oauth-step-up-authn-
challenge-01
Published: 22 March 2022
Intended Status: Standards Track
Expires: 23 September 2022
Authors: V. Bertocci B. Campbell
Auth0/Okta Ping Identity

OAuth 2.0 Step-up Authentication Challenge Protocol

Abstract

It is not uncommon for resource servers to require different authentication strengths or freshness according to the characteristics of a request. This document introduces a mechanism for a resource server to signal to a client that the authentication event associated with the access token of the current request doesn't meet its authentication requirements and specify how to meet them. This document also codifies a mechanism for a client to request that an authorization server achieve a specific authentication strength or freshness when processing an authorization 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 23 September 2022.

Copyright Notice

Table of Contents

1. Introduction

In simple API authorization scenarios, an authorization server will statically determine what authentication technique to use to handle a given request on the basis of aspects such as the scopes requested, the resource, the identity of the client and other characteristics known at provisioning time. Although the approach is viable in many situations, it falls short in several important circumstances. Consider, for instance, an eCommerce API requiring different authentication strengths depending on whether the item being purchased exceeds a certain threshold, dynamically estimated by the API itself using a logic that is opaque to the authorization server. An API might also determine that a more recent user authentication is required based on its own risk evaluation of the API request.

This document extends the error codes collection defined by [RFC6750] with a new value, insufficient_user_authentication, which can be used by resource servers to signal to the client that the authentication event associated with the access token presented with the request doesn't meet the authentication requirements of the resource server. This document also introduces acr_values and max_age parameters for the WWW-Authenticate response header defined

by [RFC6750], which the resource server can use to explicitly communicate to the client the required authentication strength or recentness.

The client can use that information to reach back to the authorization server with an authorization request specifying the authentication requirements indicated by protected resource, by including the acr_values or max_age parameter as defined in [OIDC].

Those extensions will make it possible to implement interoperable step up authentication with minimal work from resource servers, clients and authorization servers.

1.1. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Protocol Overview

Following is an end-to-end sequence of a typical step-up authentication scenario implemented according to this specification. The scenario assumes that, before the sequence described below takes place, the client already obtained an access token for the protected resource.

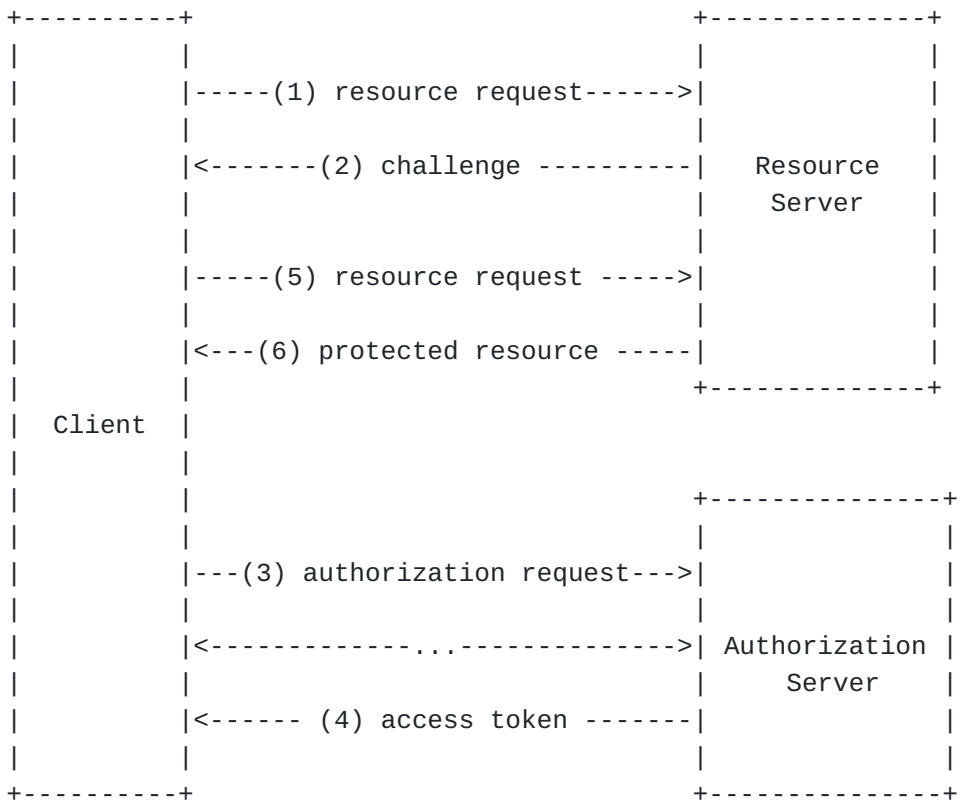

Figure 1: Abstract protocol flow

1. The client requests a protected resource, presenting an access token.

2. The resource server determines that the circumstances in which the presented access token was obtained offer insufficient authentication strength and/or freshness, hence it denies the request and returns a challenge describing (using a combination of acr_values and max_age) what authentication requirements must be met for the resource server to authorize a request.

3. The client directs the user agent to the authorization server with an authorization request that includes the acr_values and/or max_age indicated by the resource server in the previous step.

4. After whatever sequence required by the grant of choice plays out, which will include the necessary steps to authenticate the user in accordance with the acr_values and/or max_age values of the authorization request, the authorization server returns a new access token to the client. The access token contains or references information about the authentication event.

5. The client repeats the request from step 1, presenting the newly obtained access token.

6. The resource server finds that the user authentication performed during the acquisition of the new access token complies with its requirements, and returns the requested protected resource.

The validation operations mentioned in step 2 and 6 imply that the resource server has a way of evaluating the authentication level by which the access token was obtained. This document will describe how the resource server can perform that determination when the access token is a JWT Access token [RFC9068] or is validated via introspection [RFC7662]. Other methods of determining the authentication level by which the access token was obtained are possible, per agreement by the authorization server and the protected resource, but are beyond the scope of this specification.

3. Authentication Requirements Challenge

This specification introduces a new error code value for the error parameter of [RFC6750] or authentication schemes, such as [I-D.ietf-oauth-dpop], which use the error parameter:

insufficient_user_authentication The authentication event associated with the access token presented with the request doesn't meet the authentication requirements of the protected resource.

Note: the logic through which the resource server determines that the current request doesn't meet the authentication requirements of the protected resource, and associated functionality (such as expressing, deploying and publishing such requirements) is out of scope for this document.

Furthermore, this specification defines additional WWW-Authenticate auth-param values to convey the authentication requirements back to the client.

acr_values A space-separated string indicating, in order of preference, the authentication context class reference values that the protected resource requires the authentication event associated with the access token.

max_age Indicates the allowable elapsed time in seconds since the last active authentication event associated with the access token.

Below you can find an example of WWW-Authenticate header using the insufficient_user_authentication error code value to inform the client that the access token presented isn't sufficient to gain access to the protected resource, and the acr_values parameter to let the client know that the expected authentication level

corresponds to the authentication context class reference identified by myACR.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="insufficient_user_authentication",
  error_description="A different authentication level is required",
  acr_values="myACR"
```

Figure 2

If the resource server determines that the request is also lacking the scopes required by the requested resource, it MAY include the scope attribute with the scope necessary to access the protected resource, as described in section 3.1 of [RFC6750].

4. Authorization Request

A client receiving an authorization error from the resource server carrying the error code insufficient_user_authentication MAY parse the WWW-Authenticate header for acr_values and max_age and use them, if present, in a request to the authorization server to obtain a new access token complying with the corresponding requirements. Both acr_values and max_age authorization request parameters are OPTIONAL parameters defined in Section 3.1.2.1. of [OIDC]. This document does not introduce any changes in the authorization server behavior defined in [OIDC] for precessing those parameters, hence any authorization server implementing OpenID Connect will be able to participate in the flow described here with little or no changes. See Section Section 5 for more details.

The example request below indicates to the authorization server that the client would like the authentication to occur according to the authentication context class reference identified by myACR.

```
GET https://as.example.net/authorize?client_id=s6BhdRkqt3
&response_type=code&scope=purchase&acr_values=myACR
```

Figure 3

5. Authorization Response

Section 5.5.1.1 of [OIDC] establishes that an authorization server receiving a request containing the acr_values parameter MAY attempt to authenticate the user in a manner that satisfies the requested Authentication Context Class Reference, and include the corresponding value in the acr claim in the resulting ID Token. The same section also establishes that in case the desired authentication level cannot be met, the authorization server SHOULD include in the acr claim a value reflecting the authentication level of the current session (if any). The same section also states that

if a request includes thee max_age parameter, the authorization server MUST include the auth_time claim in the issued ID Token. An authorization server complying with this specification will react to the presence of the acr_values and max_age parameters by including acr and auth_time in the access token (see Section 6 for details). Although [OIDC] leaves the authorization server free to decide how to handle the inclusion of acr in ID Token when requested via acr_values, when it comes to access tokens in this specification it is RECOMMENDED that the requested acr value is treated as required for successfully fulfilling the request. That is, the requested acr value is included in the access token if the authentication operation successfully met its requirements, or that the authorization request fails in all other cases, returning unmet_authentication_requirements as defined in [OIDCUAR]. The recommended behavior will help prevent clients getting stuck in a loop where the authorization server keeps returning tokens that the resource server already identified as not meeting its requirements hence known to be rejected as well.

6. Authentication Information Conveyed via Access Token

To evaluate whether an access token meets the protected resource's requirements, the resource servers needs a way of accessing information about the authentication event by which that access token was obtained. This specification provides guidance on how to convey that information in conjunction with two common access token validation methods: the one described in [RFC9068], where the access token is encoded in JWT format and verified via a set of validation rules, and the one described in [RFC7662], where the token is validated and decoded by sending it to an introspection endpoint. Authorization servers and resource servers MAY elect to use other encoding and validation methods, however those are out of scope for this document.

6.1. JWT Access Tokens

When access tokens are represented as JSON Web Tokens (JWT) [RFC7519], the auth_time and acr claims (per Section 2.2.1 of [RFC9068]) are used to convey the time and context of the user authentication event that the authentication server performed during the course of obtaining the access token. It is useful to bear in mind that the values of those two parameters are established at user authentication time and won't change in the event of access token renewals. See the aforementioned Section 2.2.1 of [RFC9068] for details. The following is a conceptual example showing the decoded content of such a JWT access token.

Header:

```
{"typ":"at+JWT","alg":"RS256","kid":"LTacESbw"}
```

Claims:

```
{
 "iss": "https://as.example.net",
 "sub": "someone@example.net",
 "aud": "https://rs.example.com",
 "exp": 1646343000,
 "iat": 1646340200,
 "jti" : "e1j3V_bKic8-LAEB_lccD0G",
 "client_id": "s6BhdRkqt3",
 "scope": "purchase",
 "auth_time": 1646340198,
 "acr": "myACR"
}
```

Figure 4

6.2. OAuth 2.0 Token Introspection

OAuth 2.0 Token Introspection [RFC7662] defines a method for a protected resource to query an authorization server about the active state of an access token as well as to determine metainformation about the token. The following two top-level introspection response members are defined to convey information about the user authentication event that the authentication server performed during the course of obtaining the access token.

acr Authentication Context Class Reference. String specifying an Authentication Context Class Reference value that identifies the Authentication Context Class that the user authentication performed satisfied.

auth_time Time when the user authentication occurred. A JSON numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the time of date/time of the authentication event.

The following example shows an introspection response with information about the user authentication event by which the access token was obtained.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "active": true,
  "client_id": "s6BhdRkqt3",
  "scope": "purchase",
  "sub": "someone@example.net",
  "aud": "https://rs.example.com",
  "iss": "https://as.example.net",
  "exp": 1639528912,
  "iat": 1618354090,
  "auth_time": 1646340198,
  "acr": "myACR"
}
```

Figure 5

7. Security Considerations

[[TBD]]

Remember that oauth is not authN, you need a layer like OIDC to handle that part. This is not an encouragement to abuse oauth. This is about the authentication event of the user to the AS by which the access token was obtained.

8. IANA Considerations

[[TBD]]

The insufficient_user_authentication error code in the "OAuth Extensions Error" registry [IANA.OAuth.Params].

Section 6.2 for acr and auth_time as top-level members of the introspection response in the "OAuth Token Introspection Response" registry [IANA.OAuth.Params].

The acr_values and max_age WWW-Authenticate auth-params are "new" but doesn't seem like any registration is needed or possible.

9. 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>.

[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>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9068] Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021, <https://www.rfc-editor.org/info/rfc9068>.

10. Informative References

[I-D.abr-twitter-reply] Roach, A., "A reply to a specific tweet", Work in Progress, Internet-Draft, draft-abr-twitter-reply-00, 7 September 2018, <https://datatracker.ietf.org/doc/html/draft-abr-twitter-reply-00>.

[I-D.ietf-oauth-dpop] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP)", Work in Progress, Internet-Draft, draft-ietf-oauth-dpop-06, 1 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-dpop-06>.

[IANA.OAuth.Params] IANA, "OAuth Parameters", <https://www.iana.org/assignments/oauth-parameters>.

[OIDC] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", 8 November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[OIDCUAR] Lodderstedt, T., "OpenID Connect Core Error Code unmet_authentication_requirements", 8 May 2019, <https://openid.net/specs/openid-connect-unmet-authentication-requirements-1_0.html>.

[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>.

Appendix A. Acknowledgements

I wanted to thank the Academy, the viewers at home, the shampoo manufacturers, etc..

Initially (kinda) discussed at the OAuth Security Workshop 2021

A number of others already but haven't kept track...

Appendix B. Document History

[[To be removed from the final specification]]

-01

* Fixed example

* Clarified/noted that scope can also be in the WWW-Authenticate/ 401

-00

* Initial Individual Draft (with all the authority thereby bestowed [I-D.abr-twitter-reply]).

Authors' Addresses

Vittorio Bertocci
Auth0/Okta

Email: vittorio@auth0.com

Brian Campbell
Ping Identity

Email: bcampbell@pingidentity.com