Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: September 28, 2019

D. Fett
yes.com
J. Bradley
Yubico
B. Campbell
Ping Identity
T. Lodderstedt
yes.com
M. Jones
Microsoft

March 27, 2019

OAuth 2.0 Demonstration of Proof-of-Possession at the Application Layer

draft-fett-oauth-dpop-00

Abstract

This document defines a sender-constraint mechanism for OAuth 2.0 access tokens and refresh tokens utilizing an application-level proof-of-possession mechanism based on public/private key pairs.

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 September 28, 2019.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-oauth-mtls] describes methods to bind (sender-constrain) access tokens using mutual Transport Layer Security (TLS) authentication with X.509 certificates.

[I-D.ietf-oauth-token-binding] provides mechanisms to sender-constrain access tokens using HTTP token binding.

Due to a sub-par user experience of TLS client authentication in user agents and a lack of support for HTTP token binding, neither mechanism can be used if an OAuth client is a Single Page Application (SPA) running in a web browser.

This document defines an application-level sender-constraint mechanism for OAuth 2.0 access tokens and refresh tokens that can be applied when neither mTLS nor OAuth Token Binding are utilized. It achieves proof-of-possession using a public/private key pair.

1.1. Conventions and Terminology

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.

This specification uses the terms "access token", "refresh token", "authorization server", "resource server", "authorization endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", "access token response", and "client" defined by The OAuth 2.0 Authorization Framework [RFC6749].

2. Main Objective

Under the attacker model defined in [I-D.ietf-oauth-security-topics], the mechanism defined by this specification tries to ensure *token replay at a different endpoint is prevented*.

More precisely, if an adversary is able to get hold of an access token because it set up a counterfeit authorization server or resource server, the adversary is not able to replay the respective access token at another authorization or resource server.

Secondary objectives are discussed in Section 10.

3. Concept

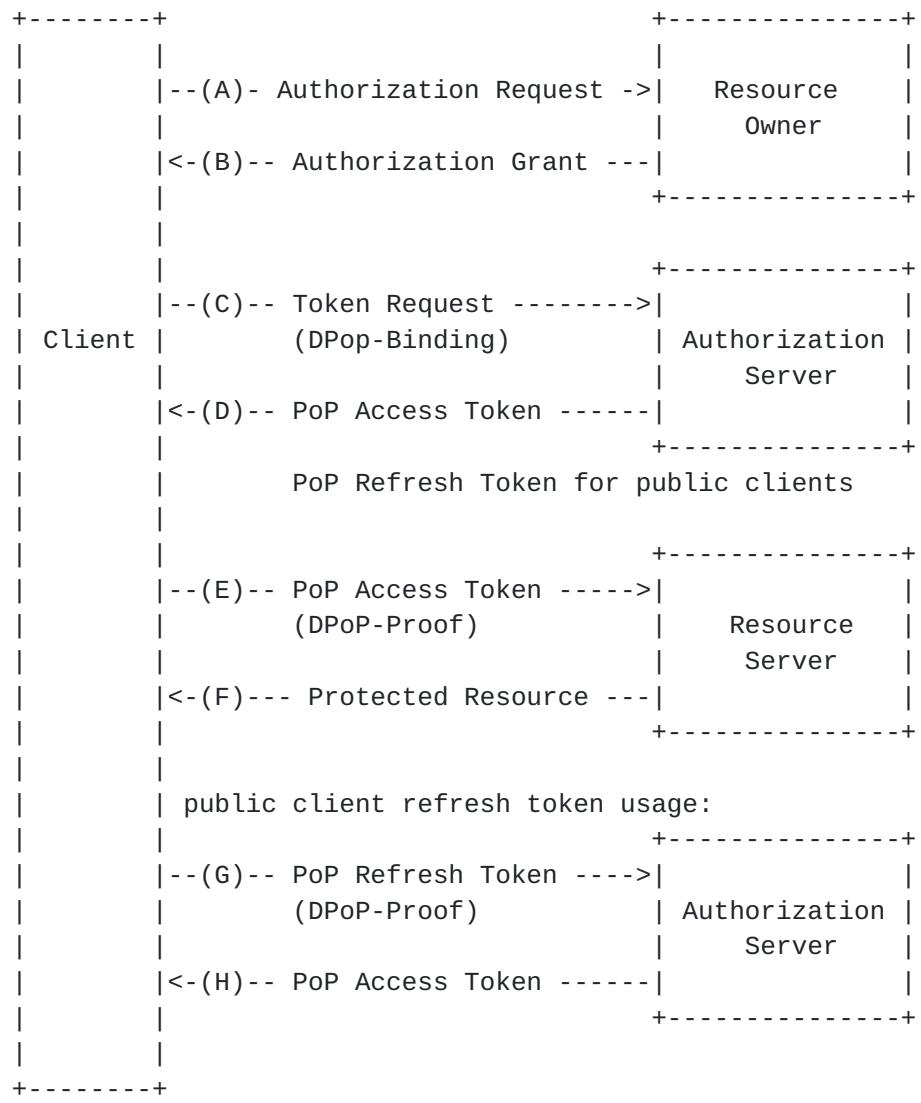

Figure 1: Basic DPoP Flow

The new elements introduced by this specification are shown in Figure 1:

- In the Token Request (C), the client proves the possession of a private key belonging to some public key by using the private key to sign the authorization code. The matching public key is sent in the same request.

- The AS binds (sender-constrains) the access token to the public key claimed by the client; that is, the access token cannot be used without proving possession of the respective private key. This is signaled to the client by using the "token_type" value "bearer+dpop". If a refresh token is issued to the client, it is sender-constrained in the same way if the client is a public

client and thus is not able to authenticate requests to the token endpoint.

- If the client wants to use the access token (E) or the (public) client wants to use a refresh token, the client has to prove possession of the private key by signing a message containing the respective token, the endpoint URL, and the request method. This signature is provided as a signed JWT.

- In the case of the refresh token, the AS can immediately check that the JWT was signed using the matching private key claimed in request (C).

- In the case of the access token, the resource server needs to receive information about which public key to check against. This information is either encoded directly into the access token, for JWT structured access tokens, or provided at the token introspection endpoint of the authorization server (request not shown).

The mechanism presented herein is not a client authentication method. In fact, a primary use case are public clients (single page applications) that do not use client authentication. Nonetheless, DPoP is designed such that it is compatible with "private_key_jwt" and all other client authentication methods.

4. Token Request (Binding Tokens to a Public Key)

To bind an tokens to a public key in the token request, the client MUST provide a public key and prove the possession of the corresponding private key. The following HTTPS request illustrates the protocol for this (with extra line breaks for display purposes only):

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded;charset=UTF-8

grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&token_type=bearer+dpop
&dpop_binding=eyJhbGciOiJSU0ExXzUi ...
(remainder of JWK omitted for brevity)
```

Figure 2: Token Request for a DPoP bound token.

The parameter "dpop_binding" MUST contain a JWT signed using the asymmetric key chosen by the client. The header of the JWT contains the following fields:

- "typ": with value "dpop_binding+jwt" (REQUIRED).
- "jwk": The public key chosen by the client, in JWK format (REQUIRED).

The body of the JWT contains the following fields:

- "http_method": The HTTP method used for the request (REQUIRED).
- "http_uri": The HTTP URI used for the request, without query and fragment parts (REQUIRED).
- "exp": Expiration time of the JWT (REQUIRED). See Security Considerations [1].
- "jti": Unique, freshly chosen identifier for this JWT (REQUIRED). SHOULD be used by the AS for replay detection and prevention. See Security Considerations [2].

An example JWT is shown in Figure 3.

```
{
    "typ": "dpop_binding+jwt",
    "alg": "ES512",
    "jwk": {
        "kty" : "EC",
        "kid" : "11",
        "crv" : "P-256",
        "x" : "usWxHK2PmfnHKwXPS54m0kTcGJ90UiglWiGahtagnv8",
        "y" : "3BttVivg+lSreASjpkttcsz+1rb7btKLv8EX4"
     }

}.{
    "jti": "HK2PmfnHKwXP",
    "http_method": "get",
    "http_uri": "https://resource-server.example.com?path=something",
    "exp": "..."
}
```

Figure 3: Example JWT for "dpop_binding" parameter.

If the authorization server receives a "dpop_binding" parameter in a token request, the authorization server MUST check that:

- the parameter value is a well-formed JWT,
- all required claims are contained in the JWT,
- the algorithm in the header of the JWT is supported by the application and deemed secure,
- the JWT is signed using the public key contained in the header of the JWT,
- the "typ" field in the header has the correct value,
- the "http_method" and "http_uri" claims match the respective values for the HTTP request in which the parameter was received,
- the token has not expired, and
- if replay protection is desired, that a JWT with the same "jti" value has not been received previously.

If these checks are successful, the authorization server MUST associate the access token with the public key. It then sets "token_type" to "bearer+dpop" in the token response.

5. Resource Access (Proof of Possession for Access Tokens)

To make use of an access token that is token bound to a public key using DPoP, a client MUST prove the possession of the corresponding private key. More precisely, the client MUST create a JWT and sign it using the previously chosen private key.

The JWT has the same format as above, except:

- The header MUST contain a "typ" claim with the value "dpop_proof+jwt".
- The header SHOULD not contain a "jwk" field.

The signed JWT MUST then be sent in the "dpop_proof" request parameter.

If a resource server detects that an access token that is to be used for resource access is bound to a public key using DPoP (via the methods described in Section 7) it MUST check that:

- a parameter "dpop_proof" was received in the HTTP request,
- the parameter's value is a well-formed JWT,

- all required claims are contained in the JWT,
- the algorithm in the header of the JWT is supported by the application and deemed secure,
- the JWT is signed using the public key to which the access token was bound,
- the "typ" field in the header has the correct value,
- the "http_method" and "http_uri" claims match the respective values for the HTTP request in which the parameter was received,
- the token has not expired, and
- if replay protection is desired, that a JWT with the same "jti" value has not been received previously.

If any of these checks fails, the resource server MUST NOT grant access to the resource.

6. Refresh Token Usage (Proof of Possession for Refresh Tokens)

At the token endpoint, public clients MUST provide a proof of possession in the same way as for access tokens.

7. Public Key Confirmation

It MUST be ensured that resource servers can reliably identify whether a token is bound using DPoP and learn the public key to which the token is bound.

Access tokens that are represented as JSON Web Tokens (JWT)[RFC7519] MUST contain information about the DPoP public key (in JWK format) in the member "dpop+jwk" of the "cnf" claim, as shown in Figure 4.

```
{
    "iss": "https://server.example.com",
    "sub": "something@example.com",
    "exp": 1493726400,
    "nbf": 1493722800,
    "cnf":{
        "dpop+jwk": {
            "kty" : "EC",
            "kid" : "11",
            "crv" : "P-256",
            "x" : "usWxHK2PmfnHKwXPS54m0kTcGJ90UiglWiGahtagnv8",
            "y" : "3BttVivg+lSreASjpkttcsz+1rb7btKLv8EX4"
        }
    }
}
```

Figure 4: Example access token with "cnf" claim.

When access token introspection is used, the same "cnf" claim as above MUST be contained in the introspection response.

8. Acknowledgements

This document resulted from discussions at the 4th OAuth Security Workshop in Stuttgart, Germany. We thank the organizers of this workshop (Ralf Kuesters, Guido Schmitz).

9. IANA Considerations

9.1. JWT Confirmation Methods Registration

This specification requests registration of the following value in the IANA "JWT Confirmation Methods" registry [IANA.JWT.Claims] for JWT "cnf" member values established by [RFC7800].

- Confirmation Method Value: "dpop+jwk"
- Confirmation Method Description: JWK encoded public key for dpop proof token
- Change Controller: IESG
- Specification Document(s): [[this specification]]

9.2. OAuth Parameters Registry

This specification registers the following parameters in the IANA "OAuth Parameters" registry defined in OAuth 2.0 [RFC6749].

- Parameter name: dpop_binding
- Parameter usage location: token request
- Change controller: IESG
- Specification document(s): [[this specification]]
- Parameter name: dpop_proof
- Parameter usage location: token request
- Change controller: IESG
- Specification document(s): [[this specification]]

9.3. JSON Web Signature and Encryption Type Values Registration

This specification registers the "dpop+jwt" type value in the IANA JSON Web Signature and Encryption Type Values registry [RFC7515]:

- "typ" Header Parameter Value: "dpop_proof+jwt"
- Abbreviation for MIME Type: None
- Change Controller: IETF
- Specification Document(s): [[this specification]]
- "typ" Header Parameter Value: "dpop_binding+jwt"
- Abbreviation for MIME Type: None
- Change Controller: IETF
- Specification Document(s): [[this specification]]

10. Security Considerations

The Prevention of Token Replay at a Different Endpoint [3] is achieved through the binding of the DPoP JWT to a certain URI and HTTP method.

10.1. Token Replay at the same authorization server

If an adversary is able to get hold of an DPoP-Binding JWT, it might replay it at the authorization server's token endpoint with the same or different payload. The issued access token is useless as long as the adversary does not get hold of a valid DPoP-Binding JWT for the corresponding resource server.

10.2. Token Replay at the same resource server endpoint

If an adversary is able to get hold of a DPoP-Proof JWT, the adversary could replay that token later at the same endpoint (the HTTP endpoint and method are enforced via the respective claims in the JWTs). To prevent this, clients MUST limit the lifetime of the JWTs, preferably to a brief period. Furthermore, the "jti" claim in each JWT MUST contain a unique (incrementing or randomly chosen) value, as proposed in [RFC7253]. Resource servers SHOULD store values at least for the lifetime of the respective JWT and decline HTTP requests by clients if a "jti" value has been seen before.

10.3. Signed JWT Swapping

Servers accepting signed DPoP JWTs MUST check the "typ" field in the headers of the JWTs to ensure that adversaries cannot use JWTs created for other purposes in the DPoP headers.

10.4. Comparison to mTLS and OAuth Token Binding

- mTLS stronger against intercepted connections

11. References

11.1. Normative References

[RFC7253] Krovetz, T. and P. Rogaway, "The OCB Authenticated-Encryption Algorithm", RFC 7253, DOI 10.17487/RFC7253, May 2014, <https://www.rfc-editor.org/info/rfc7253>.

[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>.

11.2. Informative References

[I-D.ietf-oauth-mtls]
Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual TLS Client Authentication and Certificate-Bound Access Tokens", draft-ietf-oauth-mtls-13 (work in progress), February 2019.

[I-D.ietf-oauth-security-topics]
Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", draft-ietf-oauth-security-topics-12 (work in progress), March 2019.

[I-D.ietf-oauth-token-binding]
Jones, M., Campbell, B., Bradley, J., and W. Denniss, "OAuth 2.0 Token Binding", draft-ietf-oauth-token-binding-08 (work in progress), October 2018.

[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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/info/rfc7800>.

[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>.

11.3. URIs

[1] #Security

[2] #Security

[3] #Objective_Replay_Different_Endpoint

Appendix A. Document History

[[To be removed from the final specification]]

-00

o first draft

Authors' Addresses

Daniel Fett
yes.com

Email: mail@danielfett.de

John Bradley
Yubico

Email: ve7jtb@ve7jtb.com

Brian Campbell
Ping Identity

Email: bcampbell@pingidentity.com

Torsten Lodderstedt
yes.com

Email: torsten@lodderstedt.net

Michael Jones
Microsoft

Email: mbj@microsoft.com