OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

N. Sakimura
Nomura Research Institute
K. Li
Alibaba Group
J. Bradley
Ping Identity

March 10, 2017

The OAuth 2.0 Authorization Framework: JWT Pop Token Usage
draft-sakimura-oauth-jpop-00

Abstract

This specification describes how to use JWT POP (Jpop) tokens that were obtained through [POPKD] in HTTP requests to access OAuth 2.0 protected resources. Only the party in possession of a corresponding cryptographic key with the Jpop token can use it to get access to the associated resources unlike in the case of the bearer token described in [RFC6750] where any party in posession of the access token can access the resource.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 11, 2017.

Table of Contents

1. Introduction

This document specifies the method for the client to use a proof-of-possestion token against a protected resource. The format of such token is defined in section 3 of [RFC7800].

1.1. Notational Conventions

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 RFC 2119 [RFC2119].

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

2. Terms and definitions

For the purpose of this document, the terms defined in [RFC6749] and [RFC7800] are used.

3. JWT POP Token

JWT PoP token is a JWS signed JWT whose payload is a JWT Claims Set. The JWT claims set MUST include the following:

iss The issuer identifier of the auhtorization server.

aud The identifier of the resource server.

iat The issuance time of this token.

exp The expiry time of this token.

cnf The confirmation method.

Their semantics are defined in [RFC7519] and [RFC7800].

Following is an example of such.

```
{
 "iss": "https://server.example.com",
 "aud": "https://resource.example.org",
 "iat": "1360189224",
 "exp": "1361398868",
 "cnf":{...}
}
```

Figure 1: Example of JWT PoP Token.

4. Sender Constrained Token

There are several varieties of sender constrained token. Namely:

1. CN Constrained Token

2. Client ID Constrained Token

4.1. CN Constrained Token

CN constrained token is typically used when X.509 client certificate authentication is used at the token endpoint. In this case, the constraint is expressed by including the following member at the top level of cnf claim.

cn The Common Name of the client certificate that the client used in the authorization request.

The authorization server finds the relevant CN from the X.509 client certificate authentication that is performed at the token endpoint.

```
{
"iss": "https://server.example.com",
"sub": "joe@example.com",
"aud": "https://resource.example.org",
"exp": "1361398824",
"nbf": "1360189224",
"cnf":{
  "cn": "client.example.com"
}
```

Figure 2: Example of CN Constrained JWT.

4.2. Client ID Constrained Token

The constraint in the Client ID constrained token is expressed by including the following member at the top level of cnf claim.

cid The client_id of the client that the client used in the authorization request. The combination of the "iss" of the access token and this value forms a globally unique identifier for the client.

The authorization server finds the client ID from the client ID used in the client authentication at the token endpoint.

5. Key Constrained Token

Methods to express key constraints are extensively described in the section 3 of [RFC7800]. Such cnf claim is used in the access token described in section 3 to form a key constrained token. [RFC7800] defines 4 confirmation methods.

jwk JSON Web Key Representing Public Key

jwe Encrypted JSON Web Key

kid Key Identifier

jku JWK Set URL

Following is an example of such JWT payload.

```
{
 "iss": "https://server.example.com",
 "sub": "joe@example.com",
 "aud": "https://resource.example.org",
 "exp": "1361398824",
 "nbf": "1360189224",
 "cnf":{
   "jwk":{
     "kty": "EC",
     "use": "sig",
     "crv": "P-256",
     "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
     "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkcSA"
    }
  }
}
```

Figure 3: Example of a Key Constrained JWT.

6. Resource access method

The resource server that supports this specification MUST authenticate the Client by having it demonstrate that it is the holder of the key associated with the access token being used. The confirmation method can be broadly categorized in two forms.

CN method A method leveraging on the X.509 client certificate authentication

Signature method A method leveraging the signature on the nonce. Cases with confirmation methods equal to one of cid, jwk, jwe, kid, and jku falls into this category.

6.1. CN method

Under this method, X.509 client certificate authentication at the resource endpoint is being leveraged. The resource endpoint MUST obtain the CN of the client certificate used for the authentication and MUST verify that the value of the cn member in the cnf member matches with it.

If it does not match, the process stops here and the resource access MUST be denied.

If it was valid, then the resource server MUST verify the access token. If it is valid, the resource SHOULD be returned as HTTP response.

6.2. Signature method

For this, the following steps are taken:

1. The client prepares a nonce.

2. The client creates JWS compact serialization over the nonce.

To obtain it, first create a JSON with a name "nonce" and the value being what was received in the previous step. e.g.,

```
{
        "nonce":"dcd98b7102dd2f0e8b11d0f600bfb0c093"
}
```

Then, "jws-on-nonce" is obtained by creating a compact serialization of JWS on this JSON.

3. The client sends the request to the resource server, this time with Authorization Request Header as defined in section 4.2 of [RFC7235] with the credential as follows:

```
credentials      =  "Jpop" jpop-response
jpop-response    =  at-response "," s-response
at-response      =  "at" "=" access-token; As specified by [POPKD]
s-response       =  "s" "=" jws-on-nonce; Created in the step 3.
access-token     =  quoted-string
jws-on-nonce     =  quoted-string
```

In the following example, the access token and the jws-on-nonce are represented as access.token.jwt and jws.of.nonce for the sake of brevity.

```
GET /resource/1234 HTTP/1.0
Host: server.example.com
Authorization: Jpop at="access.token.jwt", s="jws.of.nonce"
```

Figure 4: Example resouce request

4. The resource server finds the client's public key form the access token through the methods described in [RFC7800].

5. The resource server MUST verify the value of "s" of the Authorization header. If it fails, the process stops here and the resource access MUST be denied.

6. The resource server MUST verify the access token. If it is valid, the resource SHOULD be returned as HTTP response.

7. Authorization Error

If the client requests the resource without the proper authoization header, the resource server returns a HTTP 401 response with "WWW-Authenticate" header as defined in section 4.1 of [RFC7235] with the challenge as follows:

```
challenge       =  "Jpop" jpop-challenge
jpop-challenge  =  "nonce" "=" nonce-value
nonce-value     =  quoted-string
```

Following example depicts what the response would look like.

```
HTTP/1.0 401 Unauthorized
Server: HTTPd/0.9
Date: Wed, 14 March 2017 09:26:53 GMT
WWW-Authenticate: Jpop nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093"
```

Figure 5: Example error response.

8. IANA Considerations

8.1. Jpop Authentication Scheme

A new scheme has been registered in the HTTP Authentication Scheme Registry as follows:

Authentication Scheme Name: Jpop

Reference: Section 3 of this specification

Notes (optional): The Named Authentication scheme is intended to be used only with OAuth Resource Access, and thus does not support proxy authentication.

8.2. JWT Confirmation Methods

- Confirmation Method Value: "cn"
- Confirmation Method Description: CN match with the TLS client auth.
- Change Controller: IESG
- Specification Document(s): This document.
- Confirmation Method Value: "cid"
- Confirmation Method Description: Client ID Confirmation
- Change Controller: IESG
- Specification Document(s): This document.

9. Security Considerations

9.1. Certificate validation

The "cn" JWT confirmation method relies its security property on the X.509 client certificate authentication. In particular, the validity of the certificate needs to be verified properly. It involves the traversal of all the certificate chain and the certificate validation (e.g., with OCSP).

9.2. Key protection

The client's secret key must be kept securely. Otherwise, the notion of PoP breaks down.

It should be noted that JWE confirmation method is significantly weaker form of the PoP, as the resource server and the authorization server can masquerade as the client.

10. Acknowledgements

The authors thank the following people for providing valuable feedback to this document. Nov Matake (YAuth).

11. References

11.1. Normative References

[POPKD] Bradley, J., Hunt, P., Jones, M., and H. Tschofenig, "OAuth 2.0 Proof-of-Possession: Authorization Server to Client Key Distribution", March 2017, <https://tools.ietf.org/html/draft-ietf-oauth-pop-key-distribution-03>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, DOI 10.17487/RFC2617, June 1999, <http://www.rfc-editor.org/info/rfc2617>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <http://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, <http://www.rfc-editor.org/info/rfc6750>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <http://www.rfc-editor.org/info/rfc7235>.

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

[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, <http://www.rfc-editor.org/info/rfc7800>.

11.2. Informative References

[PKCE] Sakimura, N., "Proof Key for Code Exchange by OAuth Public Clients", July 2015.

[POPA] Hunt, P., Ed., "OAuth 2.0 Proof-of-Possession (PoP) Security Architecture", March 2015, <https://tools.ietf.org/html/draft-ietf-oauth-pop-architecture-08>.

[TINTRO] Richer, J., "OAuth 2.0 Token Introspection", July 2015.

Appendix A. Document History

-00 Initial Version.

Authors' Addresses

Nat Sakimura
Nomura Research Institute
Otemachi Financial City Grand Cube, 1-9-2 Otemachi
Chiyoda-ku, Tokyo 100-0004
Japan

Phone: +81-3-5533-2111
Email: n-sakimura@nri.co.jp
URI: https://nat.sakimura.org/

Kepeng Li
Alibaba Group

Email: kepeng.lkp@alibaba-inc.com

John Bradley
Ping Identity

Email: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/