OAuth Working Group N. Sakimura
Internet-Draft Nomura Research Institute
Intended status: Standards Track K. Li
Expires: September 28, 2017 Alibaba Group
J. Bradley
Ping Identity
March 27, 2017

The OAuth 2.0 Authorization Framework: JWT Pop Token Usage
draft-sakimura-oauth-jpop-04

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 the corresponding cryptographic key for 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].

Copyright Notice

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].

The same methods and JWT schema elements can be used with opaque tokens and OAuth 2.0 Token Introspection. [RFC7662]

[POPKD] can be used for a client to dynamically specify a key, or the Authorization Server can use information provided by the client at registration to provide the confirmation element.

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. DN Constrained Token

2. Client ID Constrained Token

4.1. DN Constrained Token

DN 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.

dn The Distinguished Name of the client certificate as a string that the client used in the authorization request.

The authorization server finds the relevant DN 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":{
          "dn": "cn=John Doe LLC,dc=client,dc=example,dc=com"
        }
}
```

Figure 2: Example of DN 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 a Public Key

jwe Encrypted JSON Web Key

jwkt#s256 [RFC7638] Thumbprint of a JWK using the SHA-256 hash function.

x5t#s256 [RFC7515] X.509 Certificate SHA-256 Thumbprint

jku JWK Set URL

The client provides the corresponding keys or the pointers to the authorization server as a part of the client configuration. It can be done through out-of-band methods (e.g., developper portal) or through some form of dynamic registration, etc.

Following is an example of a JWT payload containing a JWK with a raw key.

```
{
 "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 JWK Key Constrained JWT.

Following is an example of a JWT payload containing a jku URI.


```
{
 "iss": "https://server.example.com",
 "sub": "joe@example.com",
 "aud": "https://resource.example.org",
 "exp": "1361398824",
 "nbf": "1360189224",
 "cnf":{
   "jku": "https://client.example.com/keys/client123-jwks"
       }
  }
```


Figure 4: Example of a jku Constrained JWT.

Following is an example of a JWT payload containing a x5t#s256 Certificate Thumbprint of a x509 certificate. .

```
        {
         "iss": "https://server.example.com",
         "sub": "joe@example.com",
         "aud": "https://resource.example.org",
         "exp": "1361398824",
         "nbf": "1360189224",
         "cnf":{
           "x5t#s256": "w5cK0ebwmCZUYDB2Y5SlESsXE8o9yZg05O89jdNidgI"
               }
          }
```

Figure 5: Example of a x5t#s256 Certificate Thumbprint 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.

Mutual TLS method A method leveraging on the X.509 client certificate authentication of the TLS connection. cn, x5t#s256, and jku confirmation methods can be used with this access method. (The JWKS referenced by the jku MUST contain JWK with x5c certificate elements for this access method)

Signature method A method leveraging the signature on the nonce. cid, jku, jwk, jwe, and, jwkt#S256 confirmation methods can be used with this access method.

6.1. Mutual TLS acess method

DN cnf method Under this method, X.509 client certificate authentication at the resource endpoint is being leveraged. The resource endpoint MUST obtain the DN of the client certificate used for the authentication and MUST verify that the value of the dn 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 is valid, then the resource server MUST verify the access token. If it is valid, the resource SHOULD be returned as HTTP response.

x5t#s256 cnf method Under this method, X.509 client certificate authentication at the resource endpoint is being leveraged. The

resource endpoint MUST obtain the client certificate used for the authentication and MUST verify that the base64url-encoded SHA-256 thumprint of the DER encoded X.509 client certificate. The x5t#s256 member in the cnf member MUST exactly match the calculated thumbprint.

If the thumbprint does not match, access token validation fails and the resource access MUST be denied.

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

jku cnf method Under this method, X.509 client certificate authentication at the resource endpoint is being leveraged. The resource endpoint MUST obtain the client certificate used for the authentication and MUST verify that the certificate matches one of the x5c elements retrieved from the [RFC7517]JWKS. Each x5c element may contain a chain of base64-encoded certificates. The client certificate MUST only be compared with the last certificate in the chain.

If the certificate does not match one in the JWKS object, access token validation fails and the resource MUST NOT be returned.

Editor's Note: We need a reference to comparing certificates. This should probably be by string comparison of the Base64 or DER encoded formats.

If the certificate matches, 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:

STEP1: The client accesses the protected resource and gets an authorization error as in Section 7. With it, the client obtains a nonce.

STEP2: The client prepares a client nonce, "cnonce", and nounce count, "nc" as defined in section 3.2.2 of [RFC2617].

STEP3: 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. The JWS MUST contain a

kid header element if the client has more than one signing key published via JWKS URI e.g.,

```
{
        "nonce":"dcd98b7102dd2f0e8b11d0f600bfb0c093",
        "nc":"00000001",
        "cnonce":"0a4f113b"
}
```

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

STEP4: 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 STEP3. *)
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 6: Example resouce request

STEP5: The resource server finds the client's public key form the access token through the methods described in [RFC7800].

STEP6: 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.

STEP7: 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 7: 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: "dn"

- Confirmation Method Description: DN 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

o Specification Document(s): This document.

9. Security Considerations

9.1. Certificate validation

The "dn" 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.

9.3. Audiance Restriction

When using the signature method the client must specify to the AS the aud it intends to send the token to, so that it can be included in the AT.

A malicious RS could receive a AT with no aud or a logical audience and then replay the AT and jws-on-nonce to the actual server.

NOTE another approach would be to include the resource in the jws-on-nonce

9.4. Dynamic client registration elements

When a AS uses dynamic client registration it may accept software statements supplied by a federation operator. Those software statements can contain a JWKS-URI that is hosted by the federation operator or protected by a certificate provisioned from a trusted root. These methods would allow the federation operator to administratively revoke the keys at the JWKS-URI without requiring the JWKS to contain x5c elements with CA issued certificates and having to have the RS perform full certificate validation for each request.

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>.

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

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

[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>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <http://www.rfc-editor.org/info/rfc7638>.

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

[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

[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>.

[RFC7636] Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, <http://www.rfc-editor.org/info/rfc7636>.

[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/