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

Daniel Fett, John Bradley, Brian Campbell, Torsten Lodderstedt, Mike Jones
Problem Statement

- OAuth 2.0 Security BCP recommends use of sender-constrained tokens
- OAuth lacks suitable mechanism for SPAs
  - mTLS for OAuth 2.0 would cause UX issues in SPAs
  - Status of Token Binding is uncertain
Main Goal

Under the attacker model defined in [I-D.ietf-oauth-security-topics], DPoP tries to prevent token replay at a different endpoint.

If an adversary is able to get hold of an access token or refresh token because it set up a counterfeit authorization server or resource server, the adversary is not able to replay the respective token at another authorization or resource server.
Stuttgart, March 2019

First Discussions
Current Proposal

---

Client

(A) Token Request ------------------->
(DPoP Proof)

(B) DPoP-bound Access Token ---------
(token_type=DPoP)
PoP Refresh Token for public clients

(C) DPoP-bound Access Token ---------->
(DPoP Proof)

(D) Protected Resource

Authorization Server

Same proof syntax!
DPoP Proof

{
    "typ": "dpop+jwt",
    "alg": "ES256",
    "jwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "f830J3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
        "y": "x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0"
    }
},

"jti": "HK2PmfnHKwXP",
"http_method": "POST",
"http_uri": "https://server.example.com/token",
"iat": 1555555555

(Signed with matching private key.)
Token Request

POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
DPoP: eyJhbGciOiJSU0ExXzUi...
grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

Client receives **token_type=DPoP** in the authorization response iff DPoP is supported.
Resource Access

GET /protectedresource HTTP/1.1
Host: resourceserver.example.com
Authorization: DPoP eyJhbGciOiJIUzI1...
DPoP: eyJhbGciOiJSU0ExXzUi...

Access Token
Public Key Confirmation

cnf/jkt#S256 claim in the introspection response or the JWT access token:

```
{
    "iss": "https://server.example.com",
    "sub": "something@example.com",
    "exp": 1503726400,
    "nbf": 1503722800,
    "cnf": {
        "jkt#S256": "oKIywvGUpTVTyxMQ3bwIIEuudfr_CkLMjCE19ECD-U"
    }
}
```

base64url encoding [RFC7515] of the JWK SHA-256 Thumbprint [RFC7638] of the public key to which the token is bound.
Security of DPoP

- **DPoP Proof replay**
  - jti, iat, http_uri, http_method claims to avoid double use of the same claim

- **Signed JWT swapping**
  - servers must check typ claim

- **Signature algorithms**
  - none type not allowed

- **Message integrity**
  - Not guaranteed by DPoP
  - Use end-to-end TLS!
  - Bring your own data-to-sign (and add it to the DPoP proof)

→ mTLS more robust (should be used if possible)
Next Steps
Next Steps

- Individual draft → Call for adoption by WG
- Small number of open issues - no major blockers
  - Metadata
  - IANA considerations
  - Examples
  - Implicit flow
  - Error codes

**Working Examples** by Filip Skokan:
RP: [https://murmuring-journey-60982.herokuapp.com](https://murmuring-journey-60982.herokuapp.com)
OP: [https://op.panva.cz/.well-known/openid-configuration](https://op.panva.cz/.well-known/openid-configuration)