OAuth 2.0 Attestation-Based Client Authentication

Tobias Looker, Paul Bastian
Motivation

- Environments that *public* clients are operating/deployed in increasingly have primitives that can be used for client authentication examples such as:
  - App Attest on iOS for Native iOS applications
  - Play Integrity on Android for Native Android applications
- The question is how to appropriately use these capabilities to allow clients to authenticate with an authorization server?
Proposed Solution

- Extends the established framework of RFC7521 for a new form of client authentication
- Client instance obtains an attestation from client backend
- Client backend may perform any number of security checks before issuing a key-bound attestation JWT to the client instance
- Client instance authenticates towards Authorization server during a token or PAR request
- **Note** - how the client communicates with the client backend in steps 2&4 are out of scope
Key Callouts

- Proof of possession enabled client authentication method
- Can be used to authenticate the key used to bind to an access token via DPoP
- Direct mode of authentication between the client instance and the authorization server rather than a backend for front end pattern
- Avoids the client instance from having to register with the AS via DCR
Native App Example
Example - Token Request

POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTcu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A&
client_assertion-type%3Ajwt-client-attestation&
client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
eyJpc3Mi[...omitted for brevity...].
eyJpc3Mi[...omitted for brevity...].
cC4hiUPo[...omitted for brevity...].
eyJzI1NiIsImtpbGciOimtpZCI6IjIyIn0.
IjIyIn0[...omitted for brevity...].
ioJSUzI1[...omitted for brevity...]
Example - Token Request

POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTcu7iYz56a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-client-attestation&
client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
eyJpc3Mi[...omitted for brevity...].
cC4h1UPO[...omitted for brevity...].
eJyIyIn0[...omitted for brevity...].
i0iJSUzI1[...omitted for brevity...]
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTcu7iYz56a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-client-attestation&
client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
eyJpc3Mi[...omitted for brevity...].
eyJpc3Mi[...omitted for brevity...].
cyC4hiUPo[...omitted for brevity...].
eyJzI1NiIsIImtpbhc0imtpZC1IjIyIn0.
IjIyIn0[...omitted for brevity...].
I0iJSUzI1[...omitted for brevity...]

Two JWTs concatenated via a '~~' character
- Client Attestation
- Client Attestation PoP
Example - Client Assertion

eyJhbGciOiAiRVMyNTYiLCJraWQiOiAiMTEifQ.eyJpc3MiOiJodHRwczovL2NsaWVudC5leGFtcGxlLmNvbSIIsInN1YiI6Ii0wNzIlM0ZlMjL0dIY2h1M1VfODRzeG0xNjA1ODVnIiwieSI6Im1heG1kNjJ2Smx3Z2I0dEYjdGJ5dG83bMTZc0lTb1RkaWw3IiwibmJmIjoxMzAwODE1NzgwLCJleHAiOjEzMDA4MTkzODAsImNuZiI6eyJqd2siOiJteGpaU2IifQ

Client Attestation

Client Attestation PoP

Note signatures are invalid
Example - Client Attestation

```json
{
    "alg": "ES256",
    "kid": "11"
}
{
    "iss": "https://client.example.com",
    "sub": "https://client.example.com",
    "nbf": 1300815780,
    "exp": 1300819380,
    "cnf": {
        "jwk": {
            "kty": "EC",
            "crv": "P-256",
            "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
            "y": "-V4dS4UaLMgP_4fY4j8ir7c11TX1FdAgcx55o7TkSA"
        }
    }
}
```
Example - Client Attestation

```
{
    "alg": "ES256",
    "kid": "11"
}
{
    "iss": "https://client.example.com",
    "sub": "https://client.example.com",
    "nbf": 1300815780,
    "exp": 1300819380,
    ... //other claims
    "cnf": {
        "jwk": {
            "kty": "EC",
            "crv": "P-256",
            "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
            "y": "-V4dS4UaLMgP_4fY4j8ir7c17lTXlFdAgcx55o7TkcsA"
        }
    }
}
```

Key used to verify the Client Attestation PoP
Use Cases

- Potentially applicable to any OAuth 2.0 Flow that uses Client Authentication
- Concrete applications include:
  - eIDAS 2.0 usage of OpenID for Verifiable Credentials
  - Software workload authorization - enabling ephemeral software workload instances to authenticate with authorization servers without having to register with the authorization server first
Why not other approaches

- **Private key JWT Based Client Authentication**
  - Is a bearer based authentication mechanism so vulnerable to certain modes of token thief
  - A client instance would likely have to obtain a new attestation from a client backend for every AS interaction involving client authentication

- **Backend for Front Style Client Authentication**
  - Confidentiality and privacy issues for certain use-cases such as verifiable credential issuance where requests/responses would proxy through the client backend
Questions?
Links


Git Repository -> https://github.com/vcstuff/draft-looker-oauth-jwt-cwt-status-list