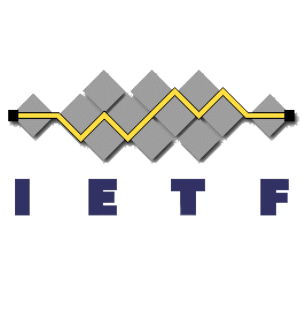
Mutual TLS Profile for OAuth 2.0

https://tools.ietf.org/html/draft-ietf-oauth-mtls-02

Brian Campbell
John Bradley
Nat Sakimura
Torsten Lodderstedt

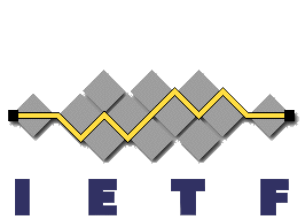
IETF 99
Prague
July 2017

What is it?



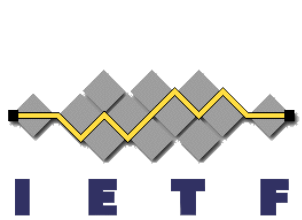
- Mutual TLS client authentication to the token endpoint
- Mutual TLS sender constrained access tokens for protected resources access

Why?



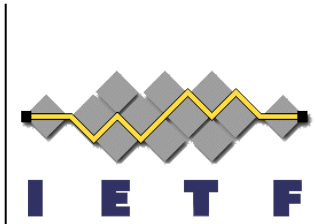
- Mutual TLS client authentication is something that's been done in practice for OAuth but we've never had a spec for it
- Mutual TLS sender constrained resources access binds access tokens to the client certificate so they can't be (re)played or used by any other entity
- Banks "need" these for server to server API use cases being driven by new open banking regulations
- Referenced by FAPI's "Read and Write API Security Profile" as a suitable holder of key mechanism
- Referenced by Open Banking API Security Profile

How Mutual TLS Client Authentication Works



- MTLS client authentication to the token endpoint
  - TLS connection from client to token endpoint is established with mutual X509 certificate authentication
  - Client includes the "client_id" HTTP request parameter in all requests to the token endpoint
  - AS verifies that the MTLS certificate is the ‘right’ one for the client
    - Trust model intentionally left open
  - Client and AS metadata

How Mutual TLS Sender Constrained Access Works



- Mutual TLS sender constrained resource access
  - Associate a hash of the certificate with the access token
  - TLS connection from client to resource is mutually authenticated TLS
    - The protected resource matches certificate from TLS connection to the certificate hash in the access token
  - New JWT Confirmation Method
    - X.509 Certificate SHA-256 Thumbprint Confirmation Method: x5t#S256
  - New Confirmation Method for Token Introspection
    - Same data as JWT x5t#S256 confirmation returned in the introspection response and checked by the protected resource
    - Requests registration of a "cnf" (confirmation) token introspection response parameter having the same semantics and format as the claim of the same name defined in RFC7800 Proof-of-Possession Key Semantics for JSON Web Tokens

JWT Confirmation

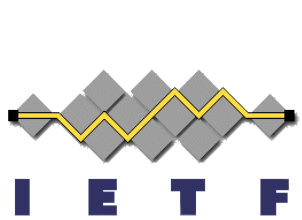
```
{
  "iss": "https://server.example.com",
  "sub": "ty.webb@example.com",
  "exp": 1493726400,
  "nbf": 1493722800,
  "cnf":{
    "x5t#S256": "bwcK0esc3ACC3DB2Y5_lESsXE8o9ltc05O89jdN-dg2"
  }
}
```

Token Introspection

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "active": true,
  "iss": "https://server.example.com",
  "sub": "ty.webb@example.com",
  "exp": 1493726400,
  "nbf": 1493722800,
  "cnf":{
    "x5t#S256": "bwcK0esc3ACC3DB2Y5_lESsXE8o9ltc05O89jdN-dg2"
  }
}
```

... and Running Code



- Recently stood up a proof of concept using COTS AS and RS products utilizing general mutual TLS support and existing configuration/customization around issuance and validation of access tokens

Next Steps

- The clock is ticking… take this thing to WGLC!