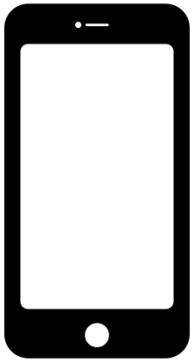
IETF 119
Brisbane
March 2024

Aaron Parecki

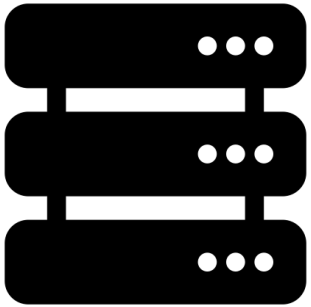
OAuth Global Token Revocation

https://datatracker.ietf.org/doc/html/draft-parecki-oauth-global-token-revocation
draft -03



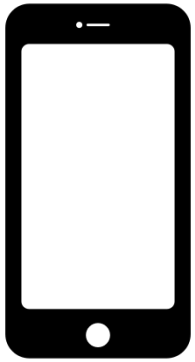
Client

“App”



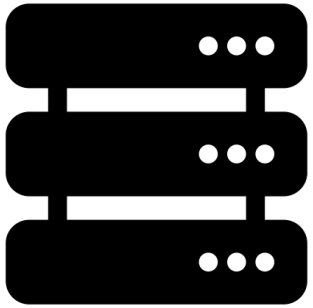
Authorization Server

“App Backend/API”



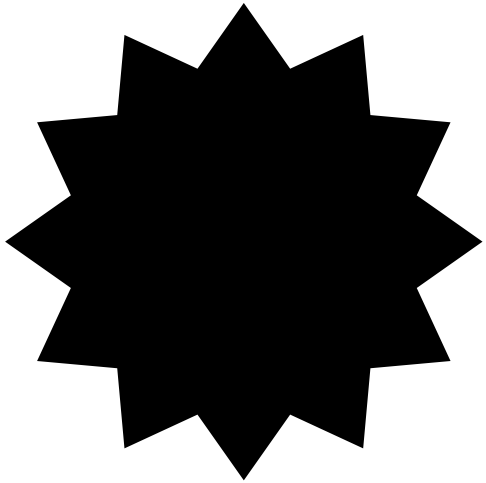
Client

“App”



Authorization Server

“App Backend/API”



Identity Provider
or
Security Monitoring Tools

Goal

Create a token revocation endpoint that provides existing applications with the shortest path to implement for interoperability.

Existing Token Revocation / Logout Standards

- RFC 7009: Token Revocation
  - Client-initiated, input is the token itself
- OpenID Connect Front-Channel Logout
  - Client-initiated
- OpenID Connect Back-Channel Logout
  - Mostly about terminating sessions
  - "refresh tokens with offline_access SHOULD NOT be revoked"
  - Tied to OpenID Connect, no clear path for SAML or other integrations
- OpenID Shared Signals Framework: CAEP / RISC
  - More of a hint or suggestion
  - Requires significant infrastructure to receive events

Global Token Revocation

Input

- Subject Identifier for Security Event Token (RFC 9493)

Authentication:

- Required, but out of scope, just like Token Introspection (RFC 7662)

Outcome:

- MUST revoke refresh tokens
- SHOULD revoke access tokens
- MUST re-authenticate the user before issuing new access tokens or refresh tokens

Global Token Revocation

https://datatracker.ietf.org/doc/html/draft-parecki-oauth-global-token-revocation-03

```
POST /global-token-revocation
Host: example.com
Content-Type: application/json
Authorization: Bearer f5641763544a7b24b08e4f74045

{
  "sub_id": {
    "format": "email",
    "email": "user@example.com"
  }
}
```

Global Token Revocation

https://datatracker.ietf.org/doc/html/draft-parecki-oauth-global-token-revocation-03

```
POST /global-token-revocation
Host: example.com
Content-Type: application/json
Authorization: Bearer f5641763544a7b24b08e4f74045

{
  "sub_id": {
    "format": "opaque",
    "id": "U1234567890"
  }
}
```

Global Token Revocation

https://datatracker.ietf.org/doc/html/draft-parecki-oauth-global-token-revocation-03

```
POST /global-token-revocation
Host: example.com
Content-Type: application/json
Authorization: Bearer f5641763544a7b24b08e4f74045

{
  "sub_id": {
    "format": "iss_sub",
    "iss": "https://authorization-server.com/",
    "sub": "af19c476f1dc4470fa3d0d9a25"
  }
}
```

Global Token Revocation

https://datatracker.ietf.org/doc/html/draft-parecki-oauth-global-token-revocation-03

HTTP response code indicates success/failure

```
HTTP/1.1 204 No Content

HTTP/1.1 400 Bad Request

HTTP/1.1 404 Not Found

etc
```

Changes from -01 to -03 since IETF 118

- Added security consideration around enumeration of user accounts
- Added security consideration for limiting scope of the authentication of the revocation request
- Added security consideration for malicious authorization servers
- Renamed property from `subject` to `sub_id` for consistency with JWT claim name defined in RFC 9493
- Added reference to `draft-ietf-oauth-status-list` as a way to revoke access tokens
- Added an appendix describing the differences between this and related logout/revocation specifications

Implementations

- Okta (in progress)
- 3 planned / on roadmap
- 3 interested

Next Steps

Working group adoption?

Authentication Options for Revocation Request

- Private Key JWT (RFC 7523)
- OAuth Access Token via Client Credentials Grant or Authorization Code Flow
- API Key

Private Key JWT Client Authentication

RFC 7523 only defines the use of a JWT as client authentication in form-encoded parameters:

```
The following example demonstrates client authentication using a JWT
during the presentation of an authorization code grant in an access
token request (with extra line breaks for display purposes only):

  POST /token.oauth2 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-bearer&
  client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
  eyJpc3Mi[...omitted for brevity...].
  cC4hiUPo[...omitted for brevity...]
```

This is fine for the OAuth Token Endpoint and similar, but not for endpoints that accept JSON

Private Key JWT Client Authentication

Can we define how to use a RFC7523 Client Authentication JWT in an HTTP header?

```
POST /example
Host: as.example.com
Content-Type: application/json
Authorization: JWT-Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.eyJpc3Mi...

{
  ...
}
```

* Also RFC7523 doesn't define a JWT `typ` for the header (JWT BCP RFC8725) so maybe we should fix that too?