The OAuth 2.0 Ecosystem

Statistics & Analysis

KU LEUVEN DistriNet

What we did

- We tested 100+ OAuth implementations
  - 94 deployed and publicly available services
  - 17 OIDC providers, 77 OAuth 2.0 API providers
  - 13 libraries and OAuth 2.0 middleware *(not included in these statistics)*

- We drew statistics over the sites and over the individual countermeasures

Supported Flows

API Providers

- 94% support Authorization Code flow
- 44% support Implicit flow
- 30% support Client Credentials flow
- 3% support Password flow

OIDC Providers

- 100% support Authorization Code flow
- 35% support Client Credentials flow
- 24% support Implicit flow
- 24% support Hybrid flow
- 6% support Device flow

Failure Rates

- Every test case in our evaluation represents one requirement in the OAuth specification
  - Of any requirement level
  - Test cases that are not applicable are not run

$$failure\ rate = \frac{test\ cases\ succeeded}{test\ cases\ run} \times 100\ \%$$

Failure Rates

API Providers

- 38.0% average failure rate (±6.9%)
  - 31% *must* failures
  - 40% *should* failures
  - 85% *may* failures

OIDC Providers

- 28.0% average failure rate (±7.0%)
  - 22% *must* failures

Failure Rates

- Popular sites do not score better (or worse)
  - Top 100 sites: 35.9% failure rate
  - Top 10000+ sites: 37.3% failure rate
- Different sites fail different tests
  - About 50% of the tests fail for less than 20% of the sites

Detailed Statistics

TLS Security

OAuth’s security relies on using TLS correctly

- All sites supported TLS 1.2 and higher
  - 11% of API endpoints did not support TLS 1.2 or higher
- 54% supported TLS 1.1 and lower
- All sites used a valid X.509 certificate

TLS Security

TLS must be used when sending sensitive information

- All sites redirect authorization requests to HTTPS
- 6% allowed insecure authorization code exchanges
- 5% allowed insecure API access

HTTP Security

Referrers may leak sensitive information

- 30% suppress the referrer

HTTP Security

Parameters must not be included multiple times

› Only 15% enforced this

Unknown parameters must be ignored

› 96% comply with this

HTTP Security

Authorization pages should not be framed

- 68% use an X-Frame-Options header
- 34% use Content Security Policy
- 26% send no header

HTTP Security

Sensitive information must not be cached

- 51% send Cache-Control and Pragma
- 6% send only Pragma
- 14% send only Cache-Control
- 29% allow caching

HTTP Security

Form POST parameters are preferred over URI query parameters

- Only 6% support *form post response mode*

OIDC requires authorization servers to support POST authorization requests

- Only 40% of OIDC servers support this

Client Authentication

Client Type

- 1% support only public clients
- 1% support confidential clients (crypto key)
- 98% support confidential client (password)
  - However, 12% do not use/require the password

Client Authentication

Authorization servers must support the *Authorization* header

- Support is mandatory, but only 69% support it
- Other sites use form POST

Proof Key for Code Exchange

Authorization servers must support PKCE

- Only 12% of API providers support PKCE
  - Mostly ignored
  - Sometimes disallowed

Proof Key for Code Exchange

For the API providers supporting PKCE:

- None required PKCE
- 33% supported *plain* PKCE
- 44% allowed very short verifiers
- 56% were vulnerable to PKCE sidestep attack[1]

[1] https://mailarchive.ietf.org/arch/msg/oauth/qrLAf3nWRt8HAFkO49qGrPRuelo/

Proof Key for Code Exchange

Half of the OIDC sites supported PKCE

- None required PKCE
- 25% supported *plain* PKCE
- 75% allowed very short verifiers
- 25% were vulnerable to PKCE sidestep attack[1]

[1] https://mailarchive.ietf.org/arch/msg/oauth/qrLAf3nWRt8HAFkO49qGrPRuelo/

Redirect URI Matching

Callback URIs must be precisely matched

› Only 48% of sites do this

Token endpoint must compare the callback URI with the one received in the authorization request

› Only 43% of sites do this

Authorization Codes

Authorization codes must only be used once

- 76% disallow code exchange
- 12% disallow code exchange and revoke previously granted access tokens
- 12% allow multiple code exchanges

Access Tokens

- Are mostly opaque (only 15% JWT)
- Are long (85% over 128 bits of entropy)
- Can often be used as URI query parameter (44%)

Refresh Tokens

- Are used by 66% of sites
- When *refresh token rotation* is used, refresh tokens must be single use
  - Of these sites, only 34% prohibited exchanging the same refresh token multiple times
  - Active refresh tokens were never revoked

Access Tokens and Refresh Tokens

If refresh tokens are used, access token lifetime should be short

- < 1 hour: 36%
- < 8 hours and > 1 hour: 27%
- < 24 hours and > 8 hours: 10%
- > 24 hours: 27%

Token Revocation

- 83% do not support token revocation *(optional)*
  - Of those that did, 42% accept revoked refresh tokens *(mandatory)*

OIDC and ID Tokens

- All sites correctly included the required claims
  - Except the “nonce” claim (18% omitted this)

- Sending the nonce parameter is mandatory for the implicit flow
  - 50% of OIDC providers do not enforce this