(The sorry)

State of the Clients

Daniel Fett, IETF 113
There is a lack of good, modern, and universal OAuth client libraries.
- follows latest security recommendations
- feels native in the language/framework
- maintained and documented

- security features (PKCE!)
- asymmetric client authentication (MTLS?)
- OAuth 2.1? FAPI?

- not tailored towards specific vendors/APIs
- not limited to certain use cases
- configurable for various feature sets
  (ideally using server metadata)
Experience in Practice

There are some good libraries, but…
Experience in Practice

... most of the time: Custom implementations!

- Hard to point devs to good libraries
  - Lack of documentation/discoverability:
    - supported features set
    - client or server?
    - supported specifications
    - security recommendations followed or not?
  - Incomplete implementations (“it works for Sign-in with Google”)
  - Many unmaintained implementations

Lack of libraries → APIs need to provide request-level description of flow → “it’s just a few requests, I can implement that myself”
Experience in Practice

- OAuth Configuration Hell™

→ without Server Metadata: Tedious process, reduced value in using libraries
The Consequences
The Consequences

- Unnecessary fragmentation
- Slow adaption of new specs
- Developer frustration
  - “several hours of research before implementing an OAuth integration”
Exhibit A: Developer Experience

Really annoyed with OAuth atm. Why are some API’s such a pain in the ass?

Why can’t companies just adhere to the OAuth spec?

Replying to @enunomaduro truly, I’ve had to do a LOT of OAuth and this makes it bearable

OAuth 2 and msal why are you such a headache ????

I am about to lose my shit over oauth

Replying to @mathew_dev @LizardSlack ... OAuth is a PITA for everyone but the customer/users :

Why is Facebook OAuth so bad

Day 14: If you weren’t sick of auth before.... working with APIs might not be for you 😖We get a really helpful step-by-step walkthrough of setting up OAuth 2.0 to hit the GitHub API.

And honestly, is there a cuter logo than Octocat? 😍

Replying to @lorenco_dan I agree in principle, but in practice, oauth is such a mess that I’m weary to trade simple mechanisms for some that are exponentially more exposed to implementation flaws.

for role-based authentication and authorization.

Instead of using Kubernetes Secrets, developers should base authentication and authorization on OIDC tokens. This means that instead of, e.g., storing a database password in a Secret resource, we should
Time and Money

- Custom implementations are expensive
  - Some API providers maintain custom OAuth implementations in several languages
  - API providers need to explain OAuth and support developers
  - Trial and error for devs to figure out supported features of AS

E.g., twitter.com expects a fully custom implementation!
Time and Money

Authorize URL

With OAuth 2.0, you create an authorize URL, which you can use to allow a user to authenticate via an authentication flow, similar to “Sign In” with Twitter.

An example of the URL you are creating is as follows:

https://twitter.com/i/oauth2/authorize?
response_type=code&client_id=M1M5R3BMVy13QmpScXkzTUt5OE46MTpjaQ&redirect_uri=https://www.example.com&scope=tweet.read%20users.read%20offline.access&state=state&code_challenge=challenge&code_challenge_method=plain

You will need to have the proper encoding for this URL to work, be sure to check out our documentation on the percent encoding.
Security

● Custom implementations are bad for security
  ○ Many opportunities for hidden security problems in custom implementations
  ○ New security recommendations are not likely to be implemented
  ○ Known anti-patterns are repeated
  ○ New security mechanisms are hard to implement

- [Li et al., 2014] 60 chinese clients, more than half vulnerable to CSRF
- [Yang et al., 2016] Out of 405 clients, 55% do not handle state (CSRF protection) correctly
- [Shebab et al., 2015] 25% of OAuth clients in Alexa Top 10000 vulnerable to CSRF
- [Chen et al., 2014] 89 of 149 mobile clients vulnerable to one or more attacks
- [Wang et al., 2013] Vulnerabilities in Facebook PHP SDK and other OAuth SDKs
- [Sun et al., 2012] 96 Clients, almost all vulnerable to one or more attacks
Let’s discuss solutions!
Proposal 1: Set a Goal

There should be defined levels of support for OAuth libraries.

- Based upon existing profiles and specs, like OAuth 2.1 or FAPI 2.0
- Or other profiles, like in OpenID Connect (+ some security requirements):

  15. Implementation Considerations
  15.1. Mandatory to Implement Features for All OpenID Providers
  15.2. Mandatory to Implement Features for Dynamic OpenID Providers

→ Provide library developers with a clear set of features to support in order to achieve interoperability.
Proposal 2: Make Metadata Mandatory

OAuth Server Metadata [RFC8414]

- enables libraries to automatically configure themselves, including
  - security mechanisms,
  - endpoints,
  - supported grant types,
- thereby drastically reducing development time and cost for clients,
- increasing the value of using libraries, and
- increasing adoption of new security features.

It should be mandatory in OAuth 2.1 and should be expected in any new OAuth ecosystem.
Proposal 3: Conformance Tests

Based upon defined profiles, provide conformance tests.

Who could do that?

Who would finance that?
Other ideas?
The End.