@justin _ _ richer

https://bspk.io/

My OAuth Wishlist


{JSON}

Consistency

- Extensions reinvent concepts
  - Device flow vs CIBA vs PAR vs UMA …
  - "resource" vs "aud" vs "claims" vs "authorization_details" …
  - JWT Assertions vs PoP vs DPoP vs OAuth-MTLS …
- Deployment assumptions
  - Registration
  - Keys and secrets

OAuth is about delegation

- Software talking to software
- Get the user involved when necessary
  - Using a web browser
- Grant types are mostly about interaction modes
  - How do I get the user involved, if at all?

Web-based interaction

- How do you get the user there?
  - Redirect
  - User code
  - Secondary device
- How do you get the user back?
  - Redirect
  - Polling
- Let's separate these concerns

Other Interaction Models

- What about communication between native apps?
- What about DID-based communication fabrics?
- What if it's not the current user you need to talk to?
- What about challenge-response crypto auth?

Who's the user?

- What if the client already knows the user?
  - Assertions or verifiable credentials
  - Existing tokens
- What if we've seen this user+client before?

When do you interact?

- OAuth forces you to choose upfront
  - Interactive flows start in the front end
- Token exchange, but need additional consent
- Need to step up access

Non-Authorization

- Calling an API, need to get a user involved
  - (Annabelle's got this covered)
- Key introduction

Ephemeral Clients and Keys

- OAuth 2 is made for web servers
  - Roughly patched for SPA and mobile
- Allow clients to create keys at runtime
- Bridge an instance of software to a trust model
  - Not a good fit for OAuth's registration model

Architectural Model

- OAuth is client-heavy
  - Everything is tied to the client
- What if we made the transaction into the core component instead?

NOT on the list:

- SOAP for OAuth
- Transport-agnostic security
- Strict backwards compatibility