Making OpenID + OAuth Simpler than the Sum of its Parts

Some early ideas we're excited about

Joseph Smarr (jsmarr@google.com)
OpenID Technology Summit
April 6, 2010

Context: "I like hybrid; make it easier!"

- Good news: Combining OpenID and OAuth ("hybrid") makes onboarding more valuable and user-friendly
  - Single consent page for AuthN and AuthZ
  - Matches user and site expectations for rich integration
  - Sites like ability to make developers pre-register (TOS)

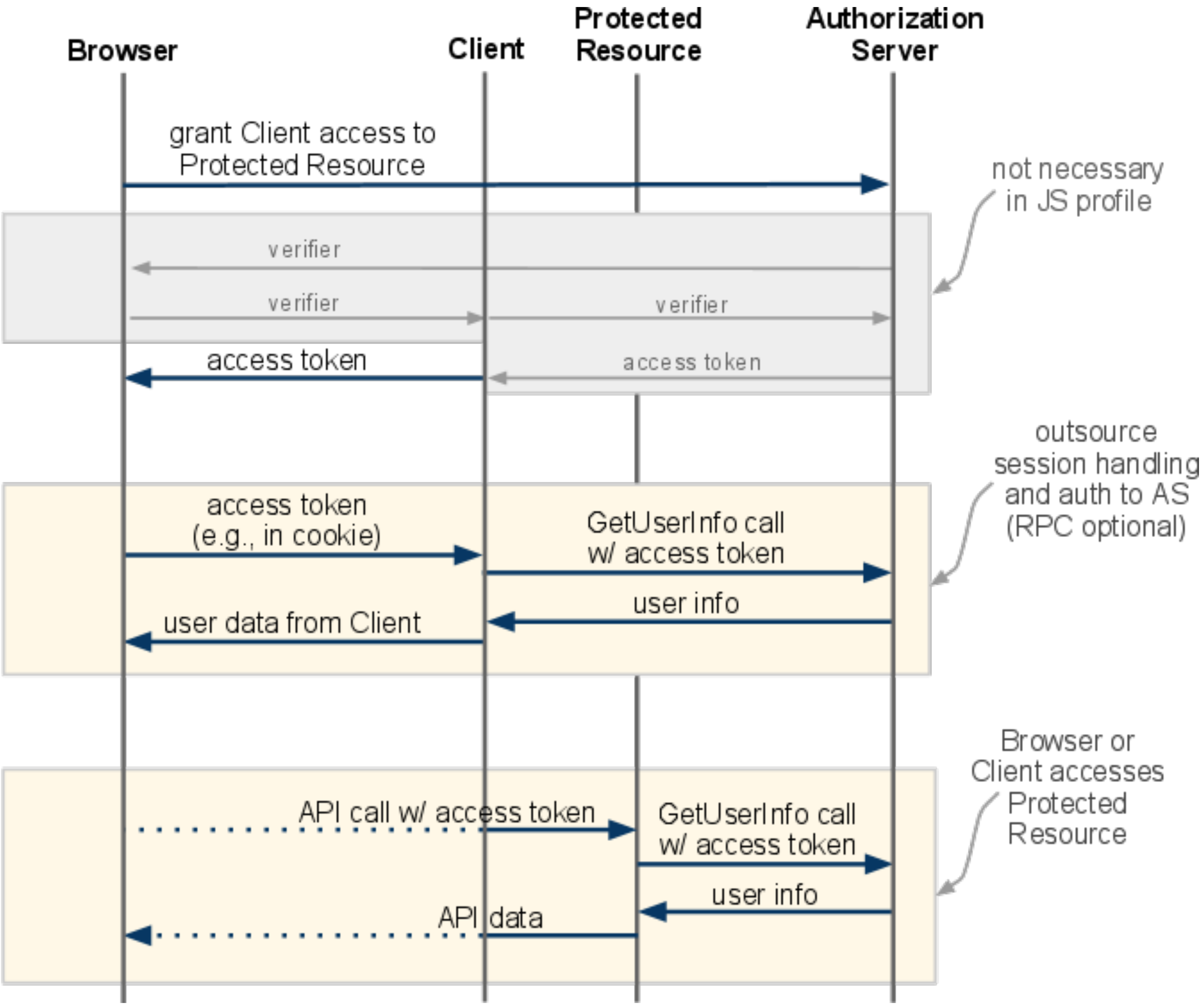
- Bad news: Both protocols are challenging to implement on their own; their composition is even harder / more confusing
  - RPs must do crypto (assertion validation, RPC signing)
  - Most complexity is server-side, and thus cannot be simplified by offering drop-in JavaScript libraries
  - RPs must handle discovery and related logic to know who to talk to and to avoid security holes

Idea: Could identity be part of OAuth?

- Evolution from OAuth 1.0 to WRAP/2.0 reflects similar goals of developer simplicity (signing requests --> bearer tokens)
- Vendor-specific hybrid protocols have an "identity RPC"
    - (twitter: users/show, facebook: users.getInfo, etc.)
- Federated challenges:
    - Who do RPs talk to when requesting user identity?
    - How do RPs verify identity assertion from OPs?
    - How do RPs ensure OPs' assertions are authoritative?
    - How can we standardize user identifiers + profile data?

An "EasyHybrid" protocol: core ideas

- Use /.well-known/auth etc. URLs to simplify discovery
- RPs can make "identity RPC" after swapping OAuth verification code for refresh/access tokens
  - Response contains OP-local stable identifier (via PoCo)
  - Returned token has RP-audience baked in
- RPs can use (OP-domain, identifier) as database key for storing local/private user data
  - OPs are authoritative for their own local identifiers
  - Server-side RPC ensures response came from OP
  - RP-audience prevents reusing tokens across RPs
- RPs can use (OP-domain, access_token) as session cookie
  - On subsequent RPCs, RP includes its RP-audience
  - Can do this on every pageview, or cache in session


Browser
Client
Protected Resource
Authorization Server
grant Client access to Protected Resource
verifier
verifier
verifier
access token
access token
not necessary in JS profile
access token (e.g., in cookie)
GetUserInfo call w/ access token
user info
user data from Client
outsource session handling and auth to AS (RPC optional)
API call w/ access token
GetUserInfo call w/ access token
user info
API data
Browser or Client accesses Protected Resource

What are we giving up for simplicity?

- Backwards-compatibility with OpenID 2.0
- Use /.well-known/auth for OP hybrid consent URL
  - Removes pre-checkid_* discovery
- OPs can only assert identities for their own domain
  - Removes post-id_res discovery
  - Eliminates security risk of cross-domain assertions
- RPs must make RPC to retrieve/validate identity assertion
  - Removes association and validation complexity
  - Have to make an RPC anyway to get access token
- Asserted identities are no longer URLs
  - But they weren't really anyway with major OP impls
  - Could use webfinger/etc for (domain, id) --> URL
- OAuth tokens are opaque (need RPC to get info anyway)
- SREG / AX
  - Move it all to GetUserInfo API, which returns PoCo data

EasyHybrid: Open questions

- Handling unregistered consumers / RPs?
    - Use anonymous/anonymous and then upgrade?
- Will we need to add static validation option back for performance reasons?
    - If so, how/when?
- What about hosted/out-sourced OP solutions
    - Which domain are identities asserted for?
    - What happens when OP wants to change providers?
- What about OpenID delegation?
    - Solve via /.well-known/auth redirect to OP?
- Backwards compatibility with existing OpenID URLs?
    - Just return them in GetUserInfo output?