SSO-next-based approach to allowing non-browser VO clients to use OAuth 2.x/OIDC

James Tocknell

AAO, Macquarie University

data central

What is this (ivoa-oauth)?

- SSO-next uses standard HTTP headers to handle auth discovery (see previous talks)
- ivoa-oauth aims to slot into the SSO-next framework a solution for OAuth 2 (and OpenID Connect which builds on it), by providing three missing parts:
  - Discovery
  - Bootstrapping client credentials
  - Setting a baseline for non-browser clients
- I'm open to changing things if it makes it easier to implement:
  - Even the name, though I'm going to use ivoa-oauth during this talk :)
- There's a mailing list discussion where I posted an initial version of this:

  http://mail.ivoa.net/pipermail/grid/2024-October/003211.html

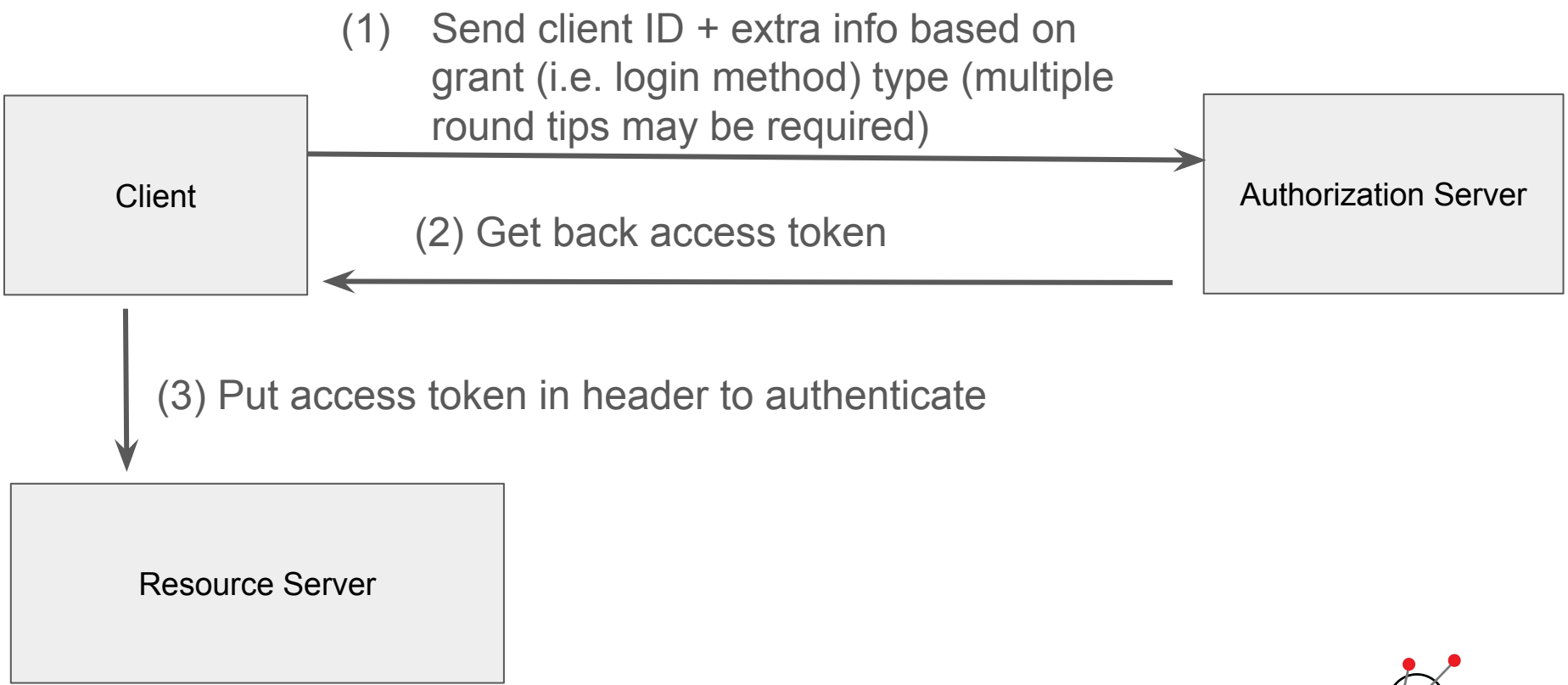
data central

Aims (as per my initial email)

1. Allow use of existing off-the-shelf OAuth 2.x/OIDC authentication providers
    a. This implies minimal changes to any of the standard flows/grants
2. Must work for non-browser clients
    a. Non-browser clients should not be required to call out a browser, nor run a web server for redirects (this covers situations where clients are running on a remote server or are otherwise unable to contact a browser directly)
3. Fit in with SSO-next
    a. This could either go into SSO-next directly, or be published as a separate note

data central

Simplified OAuth 2 system



data central

Why can’t we just use OAuth 2/OIDC?

- No standard way to go from resource server to authentication server
  - Covered by SSO-next + discovery endpoint (covered in next slide)
- Complications getting client ID in a federated system like the VO
  - Handled by the bootstrapping endpoint
- OAuth 2/OIDC in non-browser clients
  - Easiest to fix, we specify the minimum required RFCs.

data central

Discovery

- Use `www-authenticate` to specify discovery endpoint to use
  - This is following SSO-next, exact syntax TBD
  - `www-authenticate: ivoa-oauth discovery_url="https://<url of VO discovery service>"`
- Discovery endpoint (GET only, returns JSON):
  - **registration_url**: endpoint for getting client ID
  - **allowed_domains**: what domains does do the access tokens work for
  - Additional discovery metadata (either in the document, or via discovery urls) includes supported grants and additional endpoints for actual OAuth/OIDC flow
- We could either specify that the metadata be included in the JSON response, or use the OIDC discovery mechanism (uses .well-known) and/or the OAuth 2 discovery mechanism (basically same response as OIDC discovery), both could work.

data central

Bootstrapping client credentials (a.k.a “registration”)

- Use RFC 7591/OpenID Connect Dynamic Client Registration as a basis
  - OAuth 2 and OIDC are basically the same here
- There are likely complications with arbitrary clients talking directly to the client registration endpoint on your authentication server, so this is acting as a proxy to make policy decisions (e.g. allowed grant types)
- We can also move some of the many optional parameters to required
  - I suggest client_name and grant_types
- This avoids embedding in client IDs into clients, which then are shared and become meaningless; or requiring a “VO client ID” fixed string which some systems may not be able to handle

data central

Non-browser clients

- RFC 8628 a.k.a the Device Authorization Grant is the best way currently for non-browser clients to get an access token
  - This is the “scan a QR Code on your TV” grant
- Get back a URL from the authorization server, give it to the user and poll it for the access token
- The user then logs in, and the access token is available from the URL
- Part of Web Authorization Protocol (oauth) IETF working group, though not in the OAuth2 RFC nor OIDC standards, but still widely implemented it seems
- **ivoa-oauth would require RFC 8628 to be provided by authorization servers**

data central

Questions/Discussion Points

- What metadata do we want on the discovery endpoint/do we defer to existing discovery endpoints?
- What format is allowed_domains in?
- What metadata do we require for registration?
- Are there any parts of the registration process that concern you?
- Do we require OIDC, or is OAuth 2 enough (given we're not using any of the profile endpoint information)?
- Do we care about the access token contents (I'd argue no, clients should treat them as opaque)?
- Do we use Bearer or something else?
- The name ;)

data central

Useful links

- RFC 6749: The OAuth 2.0 standard
- RFC 6750: Bearer usage
- OpenID Connect Core 1.0: The base OIDC standard
- RFC 8628: The Device Authorization Grant
- RFC 9635: Grant Negotiation and Authorization Protocol (OAuth 2.x replacement, only published last month)

data central

Use of JSON

- Given the pre-existing use of JSON in the OAuth/OIDC ecosystem, JSON was chosen as the format used to avoid bringing in any additional dependencies.

data central

Avoiding use of existing discovery and client registration protocols

Given that there is substantial overlap with the existing discovery and client registration protocols, why create our own. There are two main reasons, driving mainly by the aim to allow existing authentication servers:

1. The existing providers may not implement one or both of the existing discovery protocols, and neither of the discovery protocols cover all the information that the VO Discovery Service does (and requiring the modification of an existing provider to also include the required information also has the same problem)
2. The existing providers may not implement one or both of the client registration protocols, and existing providers generally have very basic policy options (if any) around client registration. By specifying a minimal registration protocol, we avoid possible issues with existing providers.

data central

Requiring the availability of the Device Authorization Grant

Looking to the future of OAuth 2.x, where the Implicit Grant and the Resource Owner Password Credentials Grant are being removed, the Device Authorization Grant is the only standardised grant that works outside a browser. Hopefully new standardised grants are developed and supported which address the need to authenticate without a browser while being both secure and usable.