NRC·CNRC

Users, Groups, and Auth in SRCNet

SRCNet: SKA Regional Center Network

Patrick Dowler
Canadian Astronomy Data Centre

with: Brian Major (CADC), SRCNet Purple team, ThoughtWorks

General context applicable to IVOA

- clients sometimes need to authenticate to services
  - **not** browsers, web sites, portals
  - command-line tools, automated processes, batch jobs
  - astroquery, pyvo tools, etc…
  - topcat, aladdin, etc

- services and authentication
  - access to proprietary metadata and data in archives: all of DAL
  - access to project resources during research phase: VOSpace, database tables (youcat), code (docker images)
  - access to resources that inherently require permission because of resource usage: computing, write/modify to storage

SRCNet prototype: work in progress

- prototype work done by the Purple (A&A) team
  - IAM service to provide user accounts and access tokens
  - implements OpenID Connect (OIDC) portal and services
  - GMS API front end for IAM implemented by ThoughtWorks

- goal for prototype work done by CADC
  - Coral team deploy storage-inventory system provided by CADC
  - users/clients **login** to IAM and get a **token**
  - clients make requests to data management services and authenticates with http header:

    www-authenticate: bearer {token}
  - service can validate the token
  - service obtains minimal user identity info: a uuid, username, etc.
  - to be verified: service can use the token to call a GMS service to determine of the user is a member of authorized group(s)

SRCNet prototype: work in progress - using oidc-agent

- register client using device flow (once):

```
oidc-gen --iss=https://ska-iam.stfc.ac.uk --flow=device
         --scope max pdowler-ska
```

("client" info in ~/.config/oidc-agent/ – a long-lived refresh token)

- load an account (once in awhile):

```
oidc-add pdowler-ska
Enter decryption password for account config 'pdowler-ska':
```

- get or refresh access token:

```
SKA_TOKEN=$(oidc-token pdowler-ska)
```

- use the access token, for example verify it:

```
curl -s -H "authorization: bearer $SKA_TOKEN"
      https://ska-iam.stfc.ac.uk/userinfo | jq
{
  "sub": "211b77e1-686a-4116-bcff-1b2a85c442e1",
  "preferred_username": "pdowler", ...
```

SRCNet prototype: work in progress - server side

- OpenCADC libraries allow one to plug in an IdentityManager
  - code to validate authentication attempts and obtain user identity
  - prototype OIDC IdentityManager included in cadc-gms library
  - requires configuration of a “trusted” identity provider

- validate token: calls the trusted identity provider with token provided by client
- retains the credential (token) in request context for additional calls
  - currently: only send token to trusted identity provider (**server name**)
- if GMS is deployed on same server as IAM, then calls to GMS would work, otherwise: token will not be sent

code for cadc-gms library: https://github.com/opencadc/ac

SRCNet prototype: client side

- clients can call service with token:

```
SKA_TOKEN=$(oidc-token pdowler-ska)
curl -head -H "authorization: bearer $SKA_TOKEN"
      https://example.net/service/capabilities
x-vo-authenticated: pdowler
```

- tidbits:
  - my oidc-agent “client” has a (permanent) refresh token, uses it to get short-lived access tokens
  - tokens expire in 1 hour, probably IAM default and it appears to be the maximum

- simple: run code with the short-lived access token
- probably: run code with refresh token, code needs to obtain access tokens periodically using OIDC APIs

Summary

- **server side:** learned how to write plugin code to validate incoming tokens
- **client side:** can use oidc-agent to register a client and get access tokens for command line usage

- did not encounter any advice on how to a user/client is supposed to know where to get a token to access a service, and
- did not encounter any advice on how a user/client with a token should know where to send (and not send) tokens

- TODO: users/code that encounters and uses several URLs needs to know when to include the authorization header
- TODO: Let's discuss!