NRC·CNRC

NRC.CANADA.CA

Non-browser client authentication with OAuth2 tokens

Brian Major, CADC

IVOA Virtual Granada November 2020

Agenda

- New OpenID Connect and OAuth2 experience at CADC
- Server-side perspective
- Proposal using OAuth2 tokens without a browser
- How to describe use and discovery in the VO
    - (didn't quite get there)

OpenID Connect (OIDC) Support

CADC recently implemented the 'authorization server' component of OIDC. Specifically, the 'authorization code' flow.

This allows other, registered parties (projects, groups, technologies) to authenticate using CADC username/password.

We integrated the Harbor image repository (https://goharbor.io) into CADC SSO with OIDC.

CADC beta deployment

Only good for integrating browser-based applications!

OIDC in capabilities?

- A set of auth endpoints does the work of OIDC Authorization Server.

- securityMethod 'openid' exists in SSO 2.0, but how is it used?
  - Perhaps as a standardID?

- How to advertise OIDC in capabilities?
  - Not very important because OIDC has own, independent discovery mechanism based on registration.
  - Only useful as documentation

OAuth2

OpenID Connect (and SAML) are built on OAuth2

OAuth2:

- Clients need not store wide-reaching user credentials (passwords). They hold expiring tokens instead.

- Alone does not attempt to achieve interoperability:
  - Must define how discovery/bootstrapping works, how authentication happens
  - Free to solve authentication in custom manner

CADC Token Requirements

Needed access to the API of the new OIDC resource (Harbor)

- API Secured by the id_token (arbitrary)
- Implemented different 'flow' to obtain id_token from OAuth2 /authorize

Non-browser clients need access to tokens

- To access APIs of other CADC services
- Needed by cadc-tap, pyVO, vospace client tools, TOPCAT, etc..
- Why not use a 'token' flow to OAuth2 /authorize?

IVOA OAuth2 Recipe

The sweet spot for standardizing non-browser VO clients interoperating with services: **Define a VO OAuth2 recipe**

- Explain how to use SSO OAuth security method
  - `ivo://ivoa.net/sso#oauth`
- How to discover authorization server /authorize endpoint
- Define authentication processes
  - username/password (In SSO: tls-with-password)
  - HTTP Basic (In SSO: BasicAA)

Prototype Examples

Aim is to support different levels of client authentication awareness:

- Naive/legacy clients -- classic interaction with TAP
- "Lazy" clients -- don't switch to authenticated calls until needed
- Cautious/full-initialization clients -- look for authentication options upfront

Explore/prototype three different endpoints:

- /authorize (OAuth2)
- /login (authenticate username/password)
- /tap (resource server example)

Anonymous call to OAuth2 /authorize

```
REQUEST:
    HTTP GET to <base-url>/authorize?
        response_type=token&grant_type=vo
        &scope=<my-tap-ivoid>

RESPONSE:
    HTTP/1.1 401 Unauthorized
    WWW-Authenticate:
        vo-sso securitymethod="ivo://ivoa.net/sso#tls-with-certificate"
    WWW-Authenticate:
        vo-sso securitymethod="tls-with-password",
            accessURL="https://proto.canfar.net/ac/login"
```

- scope param optional

Authenticated call to OAuth2 /authorize

```
REQUEST:
    HTTP GET to <base-url>/authorize?
        response_type=token&grant_type=vo
        &scope=<my-tap-ivoid>

RESPONSE:
    HTTP/1.1 200 OK
    X-Auth-Token: <token>
    X-VO-Authenticated: <user-id>

BODY:
    <json-token-bundle>
```

- Authentication happened by some other means (eg certificate)
- This is essentially an OAuth2 CDP call
- scope param optional
- Do we need the bundle, or just the token?

OAuth2 JSON token bundle

```
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"example",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

- tokens are intended to be short-lived
- refresh tokens live longer but can be revoked

Authorize capabilities

- oauth used as a standardID

```
<capability standardID="ivo://ivoa.net/sso#oauth">
    <interface xsi:type="vs:ParamHTTP" role="std" version="0.1">
        <accessURL use="base">
            https://proto.canfar.net/ac/authorize
        </accessURL>
        <securityMethod />
        <securityMethod
            standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
    </interface>
</capability>
```

Successful (TLS) call to /login

```
REQUEST:
    HTTP GET/POST to <base-url>/login?
        userid=<userid>&password=<password>
        &scope=<my-tap-ivoid>

RESPONSE:
    HTTP/1.1 200 OK
    X-Auth-Token: <token>
    X-VO-Authenticated: <user-id>

BODY:
    <json-token-bundle>
```

- Same response as authenticated call to /authorize
- scope param optional

Login/Token capabilities

- new standardID

```
<capability standardID="ivo://ivoa.net/sso#token">
    <interface xsi:type="vs:ParamHTTP" role="std" version="0.1">
        <accessURL use="base">
            https://proto.canfar.net/ac/login
        </accessURL>
        <securityMethod
            standardID="ivo://ivoa.net/sso#tls-with-password"/>
    </interface>
</capability>
```

- Could offer a similar capability with BasicAA

Anonymous query to public TAP service

```
REQUEST:
    HTTP GET to /tap/sync

RESPONSE:
    HTTP/1.1 200 OK

BODY:
    VO-Table
```

Anonymous query to private TAP service

```
REQUEST:
    HTTP GET to /tap/sync

RESPONSE:
    HTTP/1.1 401 Unauthorized
    WWW-Authenticate:
        vo-sso securitymethod="ivo://ivoa.net/sso#tls-with-certificate"
    WWW-Authenticate:
        vo-sso securitymethod="ivo://ivoa.net/sso#oauth",
            accessURL="https://proto.canfar.net/ac/authorize"
    WWW-Authenticate:
        vo-sso securitymethod="tls-with-password",
            accessURL="https://proto.canfar.net/ac/login"
```

- **tls-with-password** and **oauth** VO behaviour need defining

Anonymous query to partially private TAP service

```
REQUEST:
    HTTP GET to /tap/sync

RESPONSE:
    HTTP/1.1 200, 401, or 404
    WWW-Authenticate:
        vo-sso securitymethod="ivo://ivoa.net/sso#tls-with-certificate"
    WWW-Authenticate:
        vo-sso securitymethod="ivo://ivoa.net/sso#oauth",
            accessURL="https://proto.canfar.net/ac/authorize"
    WWW-Authenticate:
        vo-sso securitymethod="tls-with-password",
            accessURL="https://proto.canfar.net/ac/login"
```

- 200 (OK) response has VO-Table with possible omitted content
- 401 (Unauthorized) response if query not allowed
- 404 (Not Found) response if query not allowed (but don't want to reveal that)

Successful query to auth-aware TAP service with token

```
REQUEST:
    HTTP GET to /tap/sync
    Authorization: Bearer <token>

RESPONSE:
    HTTP/1.1 200 OK
    X-VO-Authenticated: <user-id>

BODY:
    VO-Table
```

Failed query to auth-aware TAP service with token

```
REQUEST:
    HTTP GET to /tap/sync
    Authorization: Bearer <token>

RESPONSE:
    HTTP/1.1 403/404 Forbidden/Not Found
    X-VO-Authenticated: <user-id>
```

TAP Capabilities TBD

```
<capability standardID="ivo://ivoa.net/std/TAP"
xmlns:tr="http://www.ivoa.net/xml/TAPRegExt/v1.0" xsi:type="tr:TableAccess">
     <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
          <accessURL use="base">https://ws-cadc.canfar.net/youcat</accessURL>
          <securityMethod/>
          <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
          <securityMethod standardID="ivo://ivoa.net/sso#oauth"/>
     </interface>
</capability>
```

Problem: need more info to use 'oauth'

- No room in securityMethod class in XSD
- Add oauth capability (with accessURL) to this list?

Combine /authorize and /login ?

- Could consider combining functions of /authorize and /login
- Similar to the 'password' grant_type flow in OAuth2
- Would not be able to offer BasicAA for token acquisition
- Separate /login fits well with existing username/password implementations

SSO 2.0 securityMethod confusion

(or maybe just me)

	Can Identify an Interface (credentials accepted)	Can Identify a Capability (understood feature)	Should set in WWW-Authenticate?
token (proposed new)	✓	Done by HTTP(S)	?
BasicAA	✓	Done by HTTP(S)	✓
tls-with-password	token?	✓	✓
tls-with-certificate	✓	Done by HTTPS	✓
cookie	✓	Done by HTTP(S)	✓
OAuth	token	✓	✓
saml2.0	token	✓	✓
OpenID	token	✓	✓

Discovery/Bootstrap still the tough part

Discovery options for auth-aware client:

- Try the TAP service, see the auth headers (not great)
- Static auth reporting endpoint off every service baseURL (any different from capabilities?)
- Set WWW-Authenticate headers on call to /capabilities (maybe?)
- Reading service capabilities (how to convey the OAuth /authorize endpoint?)
- Out-of-band client registration? (no central authority)

Conclusions

- Using OAuth2 and HTTP to get tokens works well and leverages the security in the standard
- It’s just hard to describe how in the VO!
    - /authorize Discovery
    - Proper use of securityMethods
- Suggest a flexible, multi-path approach -- potential to mix and match different discovery and token offerings
- Shouldn't be too concerned with extra calls to negotiate tokens

- More implementation experience
- CADC prototype server implementation available very soon for client testing

SSO Updates

Security Methods need a rethink:

- Introduce the process of acquiring a token as new standardID (token?)
  - Should be able to authenticate with any interface-level securityMethod

- Should explain how to use (we use) openid, oauth, tls-with-password.

NRC·CNRC

NRC.CANADA.CA

THANK YOU

Brian Major
brian.major@nrc-cnrc.gc.ca

National Research Council Canada
Conseil national de recherches Canada

Canada