

Using OpenID/OAuth to access Federated Data Services

M. Benno Blumenthal
IRI of Columbia University

GO-ESSP 2011

10 May 2011

CMIP3

Pydap server: <http://esgcet.llnl.gov/dap/ipcc4/?thredds>

THREDDS catalog

OpenDAP service points

Basic Authentication

Cloud-accessible with basic authentication
pass-through

i.e. llnl controlled user/password access to
analysis done in the cloud with CMIP3 data

CMIP3 is cloud accessible

Example

Canadian PhD student used this IRI data library interface to do EOF, correlation and other analysis on CMIP3 model runs as well as other climate data

Now leading Climate and Agriculture group in Pakistan

Accessibility makes a difference!

Mashup Authentication

A simple data mashup: difference between two variables from two different datasets

If both datasets are access-restricted under different security realms (different userid/password), then the difference cannot be authenticated (Basic/Digest Authentication only has one set of authentication info)

OpenID

OpenID separates user identification from resource access authorization, so a federation of servers can have users with the same id, yet decide separately who gets access to their resources.

It also means resource providers can get out of the user authentication business.

This is half of the solution to the mashup problem.

Man-in-the-Middle

Modern authentication schemes (i.e. other than Basic) defend against man-in-the-middle attacks, i.e. defend against a third-party sitting in the middle of the browser and the authenticating server conversation and relaying requests while copying for nefarious reasons.

This also eliminates third-parties for good reasons. So we need to separate the good third parties from the bad ones.

More than likely, this means one must explicitly authenticate third-parties (cloud applications) as well as users.

OAuth

Mechanism to authenticate third-parties

Used by third-party apps to access Google and Facebook data, for example

Not a perfect match to our problem: built for large data provider, tiny app limit, i.e. most apps build for one data source.

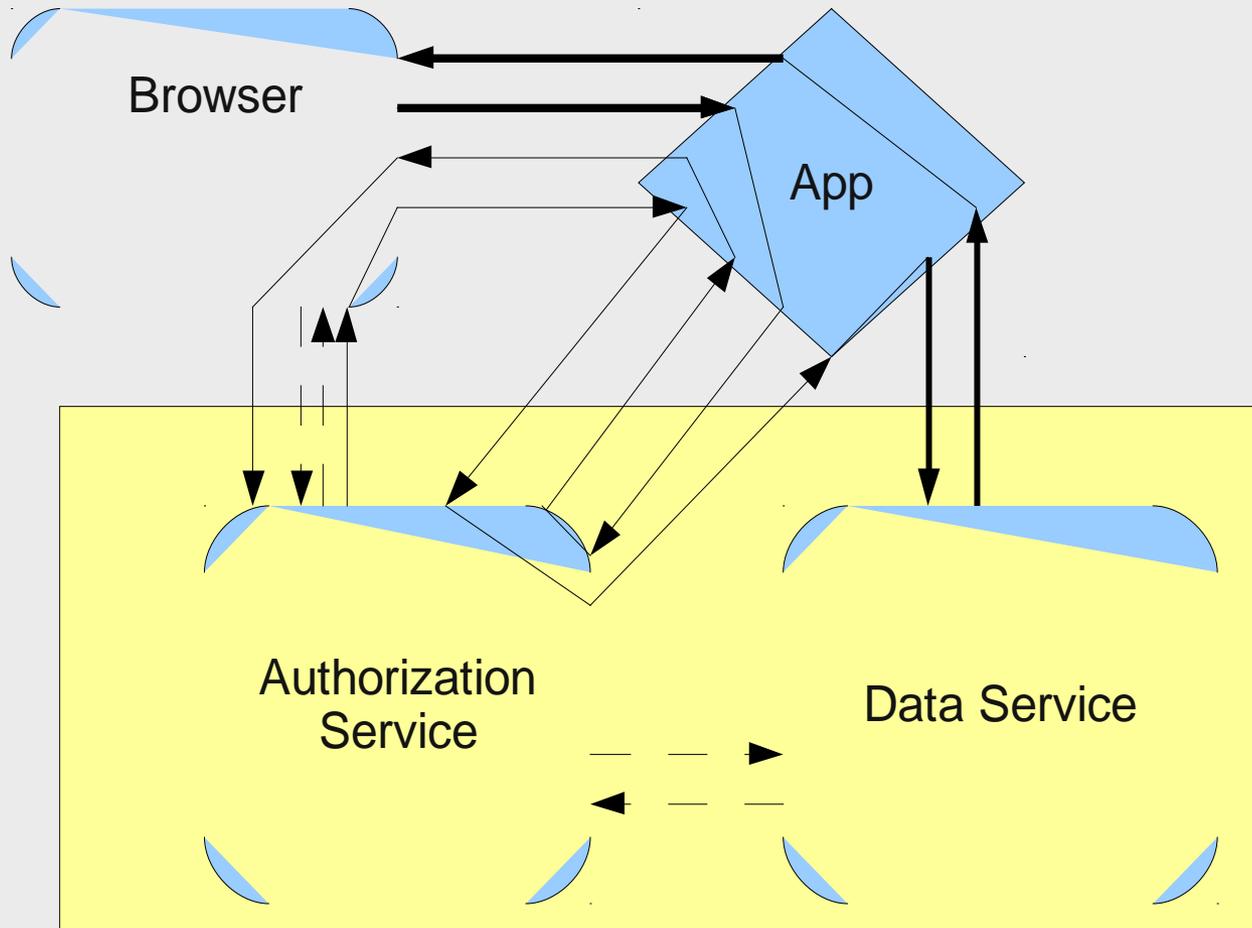
Currently missing the “refusal” part of the standard, but that could be part of OAuth 2.0, at which point it becomes “Token” Authorization (refusal does not distinguish between 2nd and 3rd parties, only the process for getting a token differs).

OAuth is token-based

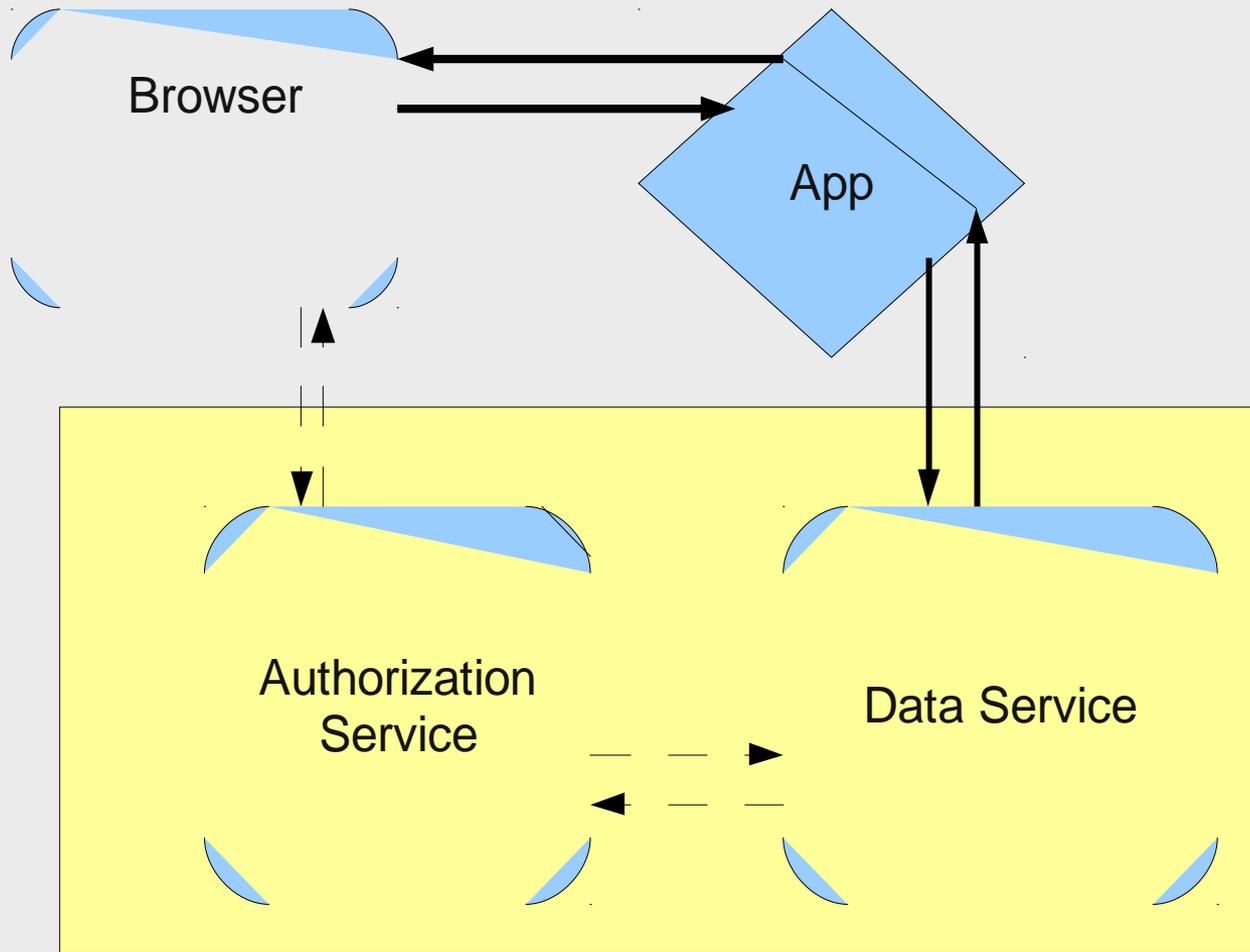
Bearer Token – possession of token is sufficient for access

MAC Token – has associated secret, i.e. token can be used over an unsecure channel

OAuth 1.0 initial access



OAuth 1.0 subsequent access



OAuth App must

- OAuth Protocol
- Identify user
- Remember authorized tokens by (user,data server)
- Probably will have pre-registered with data source authentication service

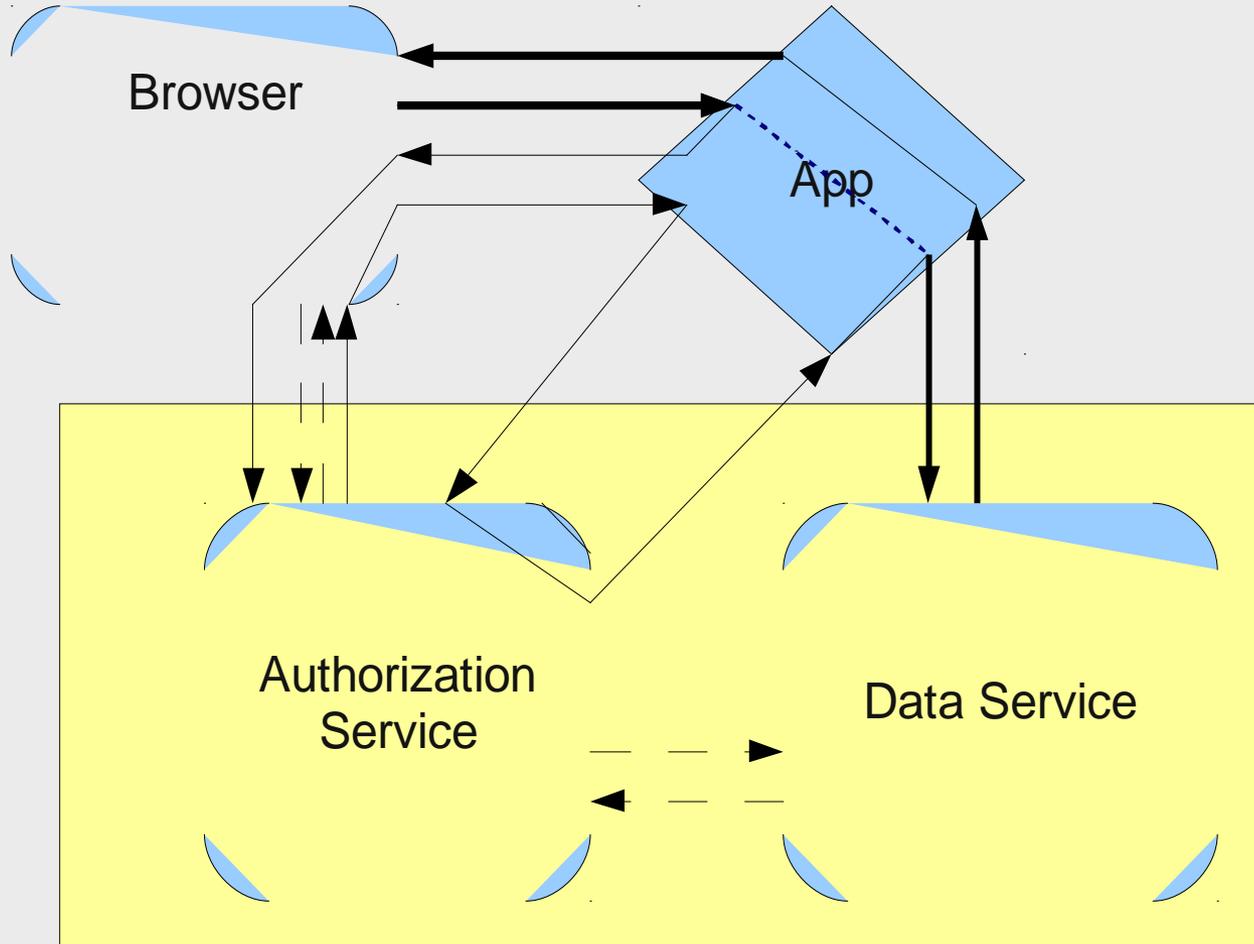
OAuth 2.0

Authorization protocol in parallel with Basic and Digest Authentication, i.e. the Unauthorized response gives the information needed to get authorization

Will probably be called “Token” or “Bearer” or “MAC” (i.e kinds of tokens) in WWW-Authenticate which will provide two endpoints for authorization

Authorization-uri: endpoint for user to identify with
token-uri: endpoint for app to identify with

OAuth 2.0 Web App Flow (*)



Federated Security for Federated Data

Authentication Service split between user identifier and data protector

Data Apps need to identify users and associated valid tokens

Data Providers need to validate both users and data apps

Need to avoid breaking anonymous/cached access

Could create a bit of a mess – when are the results of a data analysis finally released from the use-constraints of the data that went into it?