Meeting 18/1/16 to take forward task point 1 for prefs editor integration

Present: Alan, Antranig, Cindy, Dana, Giovanni, Justin, Michelle, Simon

List of actors

We would like to draw up a LIST OF ACTORS (agents) in the system to orient ourselves.

1. End user (ultimate client) sitting at a web browser, wanting to use a GPII-enabled web site.

2. Preferences consumer - owner of some resources of a web site which is to be personalised by means of GPII preferences - e.g. the ILDH.
   - May also require to edit these preferences
   - This may consist purely of static resource and this actor may not provide any code

3. GPII OAuth 2 authorization server - hosts
   - A logon UI which accepts the end user's credentials, after announcing the name of a preferences consumer on whose behalf they are being requested
   - A database associating access tokens and authorization codes with a particular end user's use of a preferences consumer

We expect that the person (agent) who wrote the Login UI exposed by the /authorize endpoint of the authorization server is the same person who implemented all the rest of the server's logic (e.g. the one managing access tokens and authorization codes), since we do not expect to share access to the user's credentials more widely.

HOWEVER, we nonetheless expect these endpoints (in the particular variant of OAuth 2 which we are planning to deploy here, the AUTHORIZATION CODE workflow) to be hosted on DISTINCT ORIGINS (in terms of HTTP's Same Origin Policy) in order to prevent leakage of credentials into the preferences consumer's origin. (Recheck and refine this point in next meeting)

Refined list of actors

2. a) The PREFERENCES EDITOR contains content which is embedded onto the preferences consumer's site (showing, for example, the panel-based UI at the top of the user wireframes)
   b) The EDGE PROXY SERVER which receives requests issued by the end user targetted at the authorization server - in particular all those listed on the GPII OAuth 2 Guide -
   /authorize - QUESTION - why does the demo not visibly use this endpoint?

/access_token

/add-preferences - QUESTION - presumably there is also a get-preferences endpoint now

c) The MULTI-PERSONALITY PROXY SERVER which is the real physical server to which the edge proxy passes HTTP requests, after adding information to the HTTP headers identifying on which preferences consumer's behalf the request was received - this proxy server unpacks the requests produced by the edge proxy server into the equivalent full requests as listed in the guide. QUESTION - is the nginx configuration language already sufficiently good that the edge proxy server could do ALL of the work of this server? If not, how much better would it need to be/what is the hardest part of this work? (e.g. unpacking a formenc request or JSON payload and inserting an extra field in it)

Workflow

Draft WORKFLOW for end user interacting with the combined system

1. User is on the site belonging to the preferences consumer. Embedded on it is a prefs editor which they have interacted with. They select an action which requires interaction with the prefs server (in the wireframes, Import or Export) Step 0
2. We enter Step 1 of the OAuth flow - the user's browser is redirected to GET <authorization-server>/authorize - Step 1a - they see a LOGIN UI (implemented by the GPII authorization server)
    a. This endpoint is NOT exposed at the same origin as the preferences consumer to avoid leakage of these credentials
    b. In the current authorization server endpoint this is exposed at the URL /login
    c. The authorization server's endpoint /login will need to be exposed by the overall cluster's proxy configuration at a distinct origin
3. The login UI redirects the user back to the consumer's space, now including an AUTHORIZATION CODE in the URL - Step 2. This can be read from the consumer's origin by client or server-side code.
4. Step 3 - this is then separately provided to the /access_token endpoint to exchange for an access token which can be used in further requests for preferences
    a. The /access_token endpoint *is* proxied by the edge server.
    b. Part of this POST body is the client secret. It appears that providing this is the key responsibility of the multi-personality server. It will be maintained securely in its configuration and fished out when it receives a matching request from the edge server.
5. The access token can be used in preferences endpoints.
    a. Question - is identification purely by access token sufficient for all of these endpoints? it appears that the authorization server can always decode the preferences consumer's id by fishing this out of its tables indexed by the token -

does this mean that the edge server's actions for all these endpoints can just be a no-op proxying?

b. Question - i) do we still need two different grant types (authorization code, and client credentials) - ii) does the presence of these different grant types have any implications at all for the implementation of the edge and proxy servers?

Meeting II notes from 19/1/16 Start Here

Questions arising from demo walkthrough on 19/1/16

We had reached the /authorize UI demonstrating step 1 of the workflow
Dana: In the case the user wants to "import" preferences, and has none, at what point do we know this when presenting them with the /authorize UI? Should the authorization server not show this interaction of there is nothing to share?

Antranig: Why is Client Credentials grant ("for adding new preferences") distinct to Authorization Code which can cover the case of writing to an existing preference set?
Answer - it is because Client Credentials is used outside the scope of an existing user account. It therefore does not need to authenticate the user.
Followon question - in the wireframes, we see "Export", and "it so happens" that the user has no GPII account. Whose responsibility is it to determine that we should enter the Client Credentials workflow rather than the Authorization Code workflow?
Further question - does this mean we need to redesign the /login endpoint of the authorization server in order to make it login/signup?

Further question - can ALL of these endpoints be declared irrelevant to the Edge Proxy Server?

Continuing with the demo - we see a UI asking the user to grant access to particular preferences a particular service. This is hosted at the /authorize endpoint -
The workflow HAS BEEN:
i) the end user was redirected to /authorize
ii) the auth server immediately issues a further redirect to /login (if they are not logged on)
iii) after logging on, the end user is redirected back to /authorize which shows the "authorization page".

Question (Dana): Which preferences get shown in the "authorization page"? It should (in UX terms) only consist of preferences which actually apply to the page in question.
Answer (Simon): This list is not constructed from an intersection of preferences that the user currently has, with the ones which the client is able to handle.

Further question: “Access” in this current implementation is ambiguous with respect to whether this is for reading or writing. The current implementation is only from the point of view of read access.

UX question here: “allow” and “do not allow” are peers in the UI but their effects are not symmetric.

Finishing with the demo - we see /authorize_callback showing the rendered preferences, using the code (plus state=RANDOM-STRING to secure the interaction from being used by a “hostile” client of the auth server which tricks the client into visiting a URL which *actually* encodes a different user id in the access token rather than their own one)

“It is used to protect against cross-site request forgery attacks.” says the GPII OAuth 2 guide about this state parameter
Section 4.4.1.8 Threat: CSRF Attack against redirect-uri

SAY WE HAD A PLAN FOR WHAT THE EDGE PROXY + MULTI PROXY SHOULD DO
ON WHAT GROUNDS COULD WE SAY THIS PLAN WAS UNVIABLE?

i) That the plan needed to proxy more endpoints
ii) That the plan allowed for some attack that the deployment of OAuth 2 was intended to defend against
iii) That the plan did not handle the user interactions that we need to support

DRAFT PLAN FOR THE PROXIES:

i) the only endpoint that we *actively* proxy is /access_token
ii) the action of the edge proxy is to decode the HTTP header, determine which “personality” of the system is being addressed (that is, which preferences consumer is receiving a request) and adds the “solution id” of that consumer to the HTTP header
iii) the request is then forwarded to the multi-personality proxy
iv) the MPP then unpacks the HTTP POST request, looks up the client’s secret from some secured persistence that it has, adds it into the body of the request, reencodes it and then forwards it to the real auth server

Can this plan be attacked on any of the grounds above?

Other endpoints (e.g. those which address the secured preferences server such as /add_preferences) are proxied without manipulating the request.

The authorization server’s “direct” endpoints (/login, /authorize) are not proxied at all.

Meeting 8th February 2016

Alan has asked:
WHAT IS THE PURPOSE OF THIS FROM THE POINT OF VIEW OF THE USER/SYSTEM?
Are we getting into “analysis paralysis”?

Simon has asked:
Oauth2 seems to be causing us a lot of trouble. We frequently have to circle around and remind ourselves what the purpose of things is. We should consider whether it’s worth checking again whether something else could solve our problems more simply, etc.

Justin/Kasper have said: All the user wants to know is that their preferences are secure.

The question is - what do we mean by secure?

What are the user’s preferences secure FROM? This implies that we start to model possible threats that we would like to guard the preferences from.

- Only explicitly authorized sites can read or write a user’s preferences
    - In addition, the user may select which preferences to share with which sites
- Having access to a user’s browsing history should not expose the ability to read or write their preferences
    - Does browsing history include client-side data management practices such as cookies, local storage?
    - What does “access” mean? Does it mean (it could mean some or all of these):
        - The “access” available to Javascript code executed from arbitrary sites
            - factors to keep in mind are inherent JS security policy (same origin, cookie access policies, browser history access policy, etc)
            - “normal” access by JavaScript code executing for other sites will not compromise access to the user’s preferences as a result of the “same origin policy”
        - “Physical” access to the browser, such as on a shared PC
            - This level of access does compromise security to the user’s preferences
    - Is this an instance of “Remember Me” security modeling, as discussed in (FE) http://stackoverflow.com/questions/244882/what-is-the-best-way-to-implement-remember-me-for-a-website, or is it something different / more complex?
    -
- Eavesdropping on the communication between the preferences client (UIO) and any server involved in the retrieval of preferences or authorization of access must not be possible (HTTPS)

High Level Threat Modeling

Refer to https://tools.ietf.org/html/rfc6819 for extended details of OAUTH threat vectors

Identifier	**Threat Vectors**
Access Token [what it is]	*Remote Acquisition* If another actor had the access token, what would they be able to do?

Meeting 9th February 2016