UMA telecon 2011-03-31

Date and Time

- WG telecon on Thursday, 31 Mar 2011, at 9:10:30am PT (time chart)
  - Skype line "C": +9900827042954214
  - US: +1-201-793-9022 (other int'l numbers) | Room Code: 295-4214

Agenda

- Roll call
- Approve minutes of 2011-03-10 and 2011-03-24 meetings
- Action item review
- News from the wider UMA world
  - SMART project
  - Fraunhofer SIT project
- Review dynamic registration materials for Prague discussion
- Drive scoped access solution
- AOB

Attendees

As of 24 Feb 2011 (post-mtg), quorum is 7 of 13.

1. Bryan, Paul
2. D’Agostino, Salvatore
3. Fletcher, George
4. Hardjono, Thomas
5. Maler, Eve
6. Moren, Lukasz
7. Morrow, Susan
8. Wolniak, Maciej

Non-voting participants:

- John Bradley
- Kirk Brown
- Kevin Cox
- Cordny Nederkoorn

Regrets:

- Domenico Catalano

Minutes

Roll call

Quorum was reached.

- Approve minutes of 2011-03-10 and 2011-03-24 meetings

The 2011-03-24 minutes are approved.

News from the wider UMA world

Lukasz reports: The SMART project hasn't been adding new functionality, instead focusing on reliability of the implementation for the IIW timeframe. The code is frozen for now. They are targeting IIW for smartam and gallerify.me announcements.
Review dynamic registration slides for Prague discussion

We reviewed the slides. UMA has two OAuth flows within it. UMA is an enhanced version of OAuth. This includes the ability to have truly autonomous parties (not just the resource owner again) asking for access, which means the resource owner may not be around.

Since our current scoped-access direction suggests that our needs for dynamic reg are reduced, maybe we should indicate this. John was hoping that we'd have brilliant ideas about dynamic registration that solve the DoS problem! Sorry that we don't.

Thomas suggests that:

- We be prepared to state whether we're planning to update the dynamic reg I-D. Currently we say yes.
- We convey UMA's vitality and future plans.
- We be prepared to state whether we're planning to contribute other I-Ds. That's the plan of record, over the next ~5 months.
- We be prepared to state any blocking issues with OAuth 2.0, since it's heading towards last call.

It's probably best to plan on revising and submitting the scoped-access spec in the next few weeks.

Eve will revise the slides a bit, and also forward to Thomas her earlier emails sent to the OAuth list from time to time.

Drive scoped access solution

We reviewed Eve's recent email (which she is going to turn into a web sequence diagram).

Actually, it's not true that being a "requesting party access token" is disruptive to OAuth; an access token in OAuth already binds a user into the mix (though OAuth assumes that it's the authorizer rather than the authorizee). So the token itself no longer represents authorization having been given until later in the process.

George asks: Can we collapse Alice-to-Bob and Alice-to-Alice flows into a single "who are you?" kind of claim, with the policy determining whether Alice-to-Alice has been satisfied? This is a mind-blowing thought. So any human being on the requesting side has to authenticate somewhere, but no longer do you need a special flow to correlate the requesting party with the authorizing user. If the authorizing user wants to consent in real time, this could be an out-of-band style of flow (similar to Liberty's Interaction Service) or it could be a user-present process, all managed entirely by the AM. The requester still may have to provide the identity of the requesting party (in a claim) independently of of any AM-side correlation.

John notes: The requester might have to present a dialog to the requesting party to say "You need to prove XYZ" and let them choose a path. This means that some details of the policy are exposed implicitly in the claims request, of course. To the extent that the claims are "policy oracle-like" in obscuring the connection between the claim asked for "what is your email address?" and the policy "email domains must be in this list: ...", this is less revealing.

Regarding the meaningful-token option, apparently the JWT/WOES work is going well in Prague this week, but we still want to wait before actually specifying things in this direction. John points out that access tokens sometimes need to be exchanged even when they're mere artifacts (e.g. when they've expired), but this is a different part of the process. We want to consider the meaningful-token option holistically when we're ready.

A:
- Req->Host HTTP request: Req makes some observable request to Host. This is unstandardized by UMA.

B:
- Host internally observes the request.

"No" branch off B:
- Host->Req (HTTP 401 error) response: Host responds to request in A with HTTP "unauthz'd" (really "unauthn'd") error and additional UMA-related AM endpoint info.

C:
- Do we have to point off to a dynamic/static client registration process here, as we have done for the host access token process in step 1? This is an opportunity for AMs to bake certain AM-and-ReqParty (vs. AM-and-ReqParty-and-AuthzUser) trust-model consequences into the otherwise relatively insignificant act of getting such an access token. Yes, we agree.
- Final step in this process: AM-Req (HTTP success) response: You have a token; try again.
- Req goes back up to A at this point.

D branch off B:
- Embedded Host-AM request-response loop begins:

E:
- Host->AM request: Host asks AM to validate token and, if valid, supply scope manifest associated with it. This is a bundling optimization, where the AM's response could branch off. Does this make sense to do? Yes, we think so.

"No, token invalid" branch off E:
- AM-Host (HTTP success) response: Token invalid.
- Host goes back up to "No" branch off B at this point.

"Yes, token valid" branch off E:
- AM-Host (HTTP success) response: Token valid; here are the scopes associated with this token.
Discussion: Would the requesting party's token be host-specific? The process of getting a requesting party access token is able to be entirely host-independent. Should it be? OAuth currently assumes that the particular host is part of the access token's binding (because of the already-tight relationship between authorization servers and resource servers).

Can we safely assume that hosts can be distinguished by domain name, such that the A-B-C process should ensure that the requestor has to inform the AM what host it was trying to access. If such tokens weren't host-specific, the meaningful-token option would get very interesting indeed, since tokens could contain really broad sets of scopes across the whole web (and could be very privacy-destroying without serious encryption protections).

If a "single host" organizationally has many domain names and endpoints at them, is it okay for UMA to make the domain name be the dividing line? If the API combines a lot of different scoped functions into a single endpoint, would it every be desirable for a single requesting party to need/use different tokens as if these were different hosts? E.g., hosts might truly be distinguished (such as in enterprise cloud API circumstances) by cloud-service-provider.com/companyname vs. companyname.cloud-service-provider.com.

At worst, perhaps we'd have to arrange for the host to tell/hint to the requestor what "host ID" (in some sense) to get a token for. For now, perhaps we can take a naive approach and then have people throw stones at it.

And there continues to be the inconvenient case of Twitter, where the endpoint is always the same regardless of the distinct user. Can we assume that the requestor already knows enough to include the necessary parameters that make the distinction, for now?

F:

- Host internally assesses the scopes returned with its knowledge of the access attempt made. If zero scopes returned, it goes to the "No" branch regardless; this means it has never been through the G branch with this requesting party.

G branch off F:

- Host-AM request message: Host registers a scope request ticket with AM, supplying a desired resource set ID and action.
- AM-Host (HTTP success) response: AM gives to Host something that can be used as scope ticket reference.
- Host-Req (HTTP 403 error) response: Host responds to request in A with HTTP "forbidden" error and additional UMA-related claims negotiation endpoint info.

H:

- Embedded Req-AM request-response loop begins, in which there is embedded a reverse claims request-response loop! This is currently outsourced to a separate spec; Claims 2.0 is our current example of how it would look, but it's incomplete in saying how the communications are fully embedded, I think. Also, we really need special Alice-to-Alice claim semantics and flows!
- Final AM-Req (HTTP success) response: No more claims needed.
- Req goes back up to A, this time wielding token that theoretically should have the right scope unless Req delays too much. When Host gets to E and F, it will discover that Req is hunky-dory.

K:

- Host-Req (HTTP success) response: Here's your protected resource.
- Req: Yay!

Next Meetings

- WG telecon on Thursday, 7 Apr 2011, at 9-10:30am PT (time chart)
- WG telecon on Thursday, 14 Apr 2011, at 9-10:30am PT (time chart)
- WG telecon on Thursday, 21 Apr 2011, at 9-10:30am PT (time chart)
- WG telecon on Thursday, 28 Apr 2011, at 9-10:30am PT (time chart) (Eve may be a bit late - can Maciej initiate call and chair for first 15 min?)