LInk to 2018 ACAMP wiki

Advance CAMP Thursday , Oct. 18, 2018

9:00am-9:50am

Banda Sea

OIDC Federation

CONVENER: Roland

MAIN SCRIBE: Eric G.

ADDITIONAL CONTRIBUTORS:
JonM

# of ATTENDEES: 45

DISCUSSION:

History: Two different OIDC Fed drafts were presented, Roland/Andreas rationalized the two drafts. This has been done (directly before TechEx, as it happens). Purpose of this presentation is to discuss the combined draft. So even if you've read Roland's original draft there are changes.

Current draft: https://storage.googleapis.com/openid-connect/oidcfed-05.html

Roland's slides

- Cornerstones
    - Trust model
    - Every entity has a unique ID.
        - In SAML you can use entity ID to look up SAML metadata, this is a convention. In OIDC this draft formalizes that (entity id leads to Entity Statement.

    - Uses JSON and JWT (Signed)
        - Trust via signature, not TLS
    - Distributed control of the federation information. E.g., Fed operator can provide some control of the metadata trust model, but is not required to control all of the metadata for all participants.
        - supports eventual growth
- Shared information is an Entity Statement.
    - Signed JSON payload describing the provider. Allows for a single entity to be both provider and client. Names “openid” as part of the metadata (e.g., “openid-client”) specifically to allow for non-openid providers/clients to be described in the entity statement.
    - Entity is simply a broad term, it can describe:
        - The federation
        - A member organization
        - An RP submitted by a member organization
    - Contains subject key material and authority hints.
        - Authority hint is esp. for distributed control use cases, where fed op and entity may each provide their own entity info that may have a (typically hierarchical) relationship.
- Trust model focuses on needing to define a common third party that both parties trust (either directly or through a hierarchical trust path)
    - A “Trust Path” in this draft is a sequence of signed JWTs
    - This differs some from the present OIDC model.
        - In SAML, entities have entityIDs even in the absence of having connected to a peer.
        - In OIDC, your entity ID effectively includes information from the peer you have connected with; the peer is effectively anonymous until you have dynamically registered with it. There is no “implicit” or “predefined” name for you or your peer in the SAML entityID sense.
        - Question about how well the OIDC Fed draft maps to this “anonymous” dynamic registration process.
            - Some discussion that this may work like SAML Federation-present peers (akin to OIDC Fed) vs. bilateral agreements (more like “standard” OIDC).
    - Static vs. dynamic registration

        - You could build a federation based on static registration, but that is not necessarily preferable. Goal here is to use dynamic registration, but still be able to leverage some form or trust model/trust anchor
        - Static registration would also require that all sites in the federation support registration in the same way, including potentially the specific toolsets used, or the need for everyone to need to log in to each location's local management console (a la registering with Google for client credentials) to make the integrations work.
    - Mobile clients
        - Roland makes the assumption (assertion?) that the user of the mobile client is likely a single user and not going to change, and thus discovery OF THE OP is a "one time" process. You don't need an ongoing discovery process.
            - Some discussion about how true that is, but general agreement that this is probably true per app on a mobile device.
        - App Auth libraries generally appear to support direct configuration of the OP metadata OR discovery.
            - (Discussion of the earlier point on entity IDs not existing until registration is completed)
    - Trust anchor skepticism
        - How interoperable is this all going to be? This is layering potential use of trust on top of OIDC. Banks use certs. Google may not support this. But will it ever be a common solution that works outside of R&E?
        - OIDC Fed isn't targeting competition with Google/FB/etc.
        - Communities like Health Care really need something like this. So should scale beyond R&E, even if it does not become the base Google/FB use cases.
        - Goal is to get support into the software, even if individual OPs don't support it. (E.g., MS client products might support OIDC Fed whether or not MS Azure participates in the trust anchor).
        - This is why it's being done in the OIDC WG arena, to make it part of a general spec.
        - More general skepticism: Entirely possible that no vendors will adopt this. Potential desire for proxy service breakout session. :)
    - Why is this an OIDC federation and not an OAuth federation?
        - Could be used as an OAuth federation.

        - However, OIDC is more similar to SAML than to general OAuth
        - OAuth is closer to the OP
            - Though perhaps UMA (User Managed Access) is a counter example of this viewpoint
    - How different is running a trust anchor service from running a traditional CA?.
        - (Question was asked but not directly answered. May be covered in the entity statement management section that follows).
- Entity statement management
    - Entity's managed metadata references a "higher level" entity.
    - You end up with an entity statement tree.
        - When is the entity tree interrogated?
        - At the time the client (or OP) is trying to determine the information about the "peer" entity.
    - Entity statement tree can have multiple trust anchors (multiple "higher level" entities).
        - A given entity may not trust every trust anchor and may filter the entity statements based on which trust anchors it will accept statements from.
    - AuthorityHints reference from the leaf back up to the parent node, every node always has a signing key. This is very similar to a certificate trust chain in terms of validation through a trust chain, you're just validating entity statements rather than actual certificates.
    - "Flattening" is the process of making a single entity statement out of the hierarchy of statements in an entity statement tree.
        - Subordinate can never overwrite what a superior has specified
        - Subordinate can specify something more restrictive that the superior.
            - (Proper subsets)
        - Gave an example of how flattening works.
        - Talked about how the flattening only allows or constrains, but does not explicitly deny.
        - Also means that is you have multiple superiors, then different peers would see different flattened metadata version depending on which trust anchors they trust.
        - There is no context: Any json tag can be added in the subordinate if it is not scoped by the superior.
        -

    - The entity statement tree implies that each entity that manages metadata would need its own “federation manager” service.
        - Or possibly the federation manager could be externalized, and the location simply signs the resulting entity statement.
- Choosing federations
    - If an RP and an OP use different trust anchors (even if there is overlap between them), they may come to incompatible conclusions about things like supported encryption mechanisms.
    - Difference in models (Roland vs Andreas):
        - In Andreas’ view, there is no “dynamic registration” per OIDC. The RP and OP each join their federation(s), and that is how they “register”. They call this “implicit registration”.
        - Roland’s model still uses explicit registration (which appears to mean the flow still follows the existing OIDC dynamic registration process, but impacted by the information in the trusted metadata)
        - Currently the model includes both options, and there would likely be profiles that leverage one, the other or both.
        - This implies that an OP might only support implicit registration and an RP only support explicit registration, so two members of the same federation may be explicitly unable to communicate.
        - Roland’s expectation is that each federation would pick one mode, it wouldn’t support both.
        - The main difference between implicit and explicit registration is that with explicit registration the RP and OP can communicate between each other how they are resolving metadata conflicts (by expressing the decision in the registration). With implicit registration both can make decisions, but they can’t establish a resolution between them.
            - What practical problems are **not** solved by implicit registration? (I.e., how much does it really matter that implicit registration loses some of this information).
        - Discussion of complexity of supporting both. Agreement that it could potentially work, but depends on exactly how the statements are crafted (e.g., “low vs. high” interop levels might make this easier to digest than other approaches).
    - Comparison to eduroam
        - eduroam is widely adopted, is

        - Is it actually a requirement that “metadata” and “registration” be simple to make things work at scale?
- Revocation
    - How long should the entity statement be good for? How to refresh signing?
    - Flow discussion
        - Implicit registration
            - You get an authn request request
            - You go off and review metadata
            - If accepted you allow registration (implicit registration)
            - Generate the claims
            - Next Authn request is evaluated based on cache time of the entity statements.
        - Explicit registration
            - More complicated
            - Gets authn
            - Check entity statement
            - Create the OAuth (internal) registration
            - Generate the claims
            - On Next Authn request, OP can’t just leverage the metadata, because the explicit registration may contain agreement information that goes beyond the detail in the signed entity statements, so just interrogating the entity statements is insufficient.
            - May require the RP/OP to un-register the peer and re-register as if it were a new peering request
            - May be necessary for RP to indicate a validity period as part of the registration (how long the registration has to be accepted for interop to work).
            - Keys in Explicit registration flows
                - The communication with explicit registration should use different keys than those used to sign metadata.
                - OIDC has a mechanism for rotating keys, but the key rotation in OIDC is based on just going and grabbing the new keys from a known endpoint (no signing). This is triggered by a JWT coming in that was signed with a key id that the recipient has never seen.

        - Recommendation/goal is to have the existing or new key to be signed by the key that's in metadata.
        - What is the purpose of allowing for the constant key registration in the traditional OIDC flow?
            - Explicit registration is basically trying to be backwards compatible with the traditional OIDC key management flow
            - Some OPs in practice do rotate keys very aggressively; e.g., every hour or two.
        - Note that in SAML the use cases for changing keys were no included in the spec; including the operational model expectations in the spec helps make the spec more cohesive.
        - Tracking keys of the root of the trust anchor is not currently spelled out in the spec.
        - This rotation is relevant for changing the keys within the protocol. But if you add new capabilities (e.g., new cryptographic mechanism), the OIDC flow doesn't support that. Requires either re-registering, or registering as a new entity.
    - OIDC re-registration is not a well defined thing. The criteria under which an existing explicitly registered entity should be added vs replace an existing registration is not well defined.
    - On the other hand, registering a brand-new entity likely breaks all existing user profiles, because users are scoped to the OIDC entity. New entity=new "scope" (from a SAML perspective).

ACTIVITIES GOING FORWARD / NEXT STEPS:

- Is this relevant for the OIDCFed or OIDC R&E groups?
    - This is not specific to R&E, so OIDF AB/Connect WG is where it should happen.
        - openid-specs-ab@lists.openid.net
        - https://openid.net/wg/connect/

=====

Note: please be sure to link to or attach any key resources from this breakout session