Rework Headscale’s OpenID Connect

This document aims to discuss some of the shortcomings in Headscale OpenID Connect implementation and how they can be rectified. Issues range from making it hard to maintain to not working as intended.

The OIDC support is currently described as “experimental”. The goal would be to have this as a full feature, and to get there, we can make breaking changes.

Problem[1]

OpenID Connect support was added to Headscale quite early in the project (#126) and before maintainers were comfortable with requiring design docs, tests and expectations of maintenance of a feature.
The result of this was that the initial implementation was made to work with one OIDC, the OIDC of the contributor, with little to no certain support of other providers.

This quickly changed as we got more contributions, big and small, that would add support for more things. The main issue was that there was no clear end goal to where the OIDC should end up and how Headscale should work with it. It did not help that neither of the maintainers had a good overview of the way OIDC worked.

Please also see the open issues and pull requests related to OIDC.

Proposal

We need to get OIDC back on track, and to do that we need to establish what we want to support, and what we want to support right now. It is likely that this journey will first take a few steps back to get some things right and then open up for extending it in the future.

In addition, for certain things, we should revert to having the same behaviour as tailscale.com.

Following is an outline of what I think needs to be done.

Non-goals

As part of the initial work, the only goal should be to support OIDC in its “purest” functional form. The criteria for this is that it passes our established integration tests with MockOIDC, potentially with some extensions if we see fit.

[1] Please note that we are happy for all the contributions, and the issues that arrised is the fault of the maintainers for not being critical or having a clear view of the feature at the time. The main goal of this section is to outline how we got there.

This means that any out-of-spec features will not be supported for the time being.

Extend User table

Currently the user table only holds the username (Name) of the user, it should be extended to hold additional user information provided by the OIDC, where the added benefit is that we can display user info nicer in the UI clients:

DisplayName - Full name of user if provided
Email - email of user if provided
OIDCIdentifier - unique identifier from OIDC (I believe this is the “sub” field from UserInfo)
Provider - registration method
ProfilePictureURL - link to profile picture

Any OIDC login should set these fields to ensure they are kept up to date.
Headscale users that use CLI based login can be given API/CLI to set these fields.

The option to strip domains from domain from emails to create users will be removed.
The existing Name field could be set to “desired_username”, which is what will be used in the Policy.

ProfilePictureURL should be set if available, otherwise fallback to gravatar with the email.

Testing

Currently, the login flow is tested with MockOIDC as part of our integration tests, this should be the passing criteria for no regressions. This test should also be extended to verify that the fields in the extended User table are set appropriately.

In addition, it would be great if the tests could be extended to run against Dex and/or Keycloak[2] to test with proper OIDC providers that we know our users use.

Non-standard OIDC

There are a couple of open PRs for “non-standard” behaviour;

- Google Workspace does not follow the group part of the spec
- Keycloak has a structured “group/subgroup” scheme

And there are probably other edge cases with different providers too that are not yet raised.

[2] Automatic Keycloak for integration testing:
https://www.baeldung.com/spring-boot-keycloak-integration-testing

These PRs add custom code and custom config options intertwined with the other code making it very hard to maintain, and there is no upper limit for how many special cases we need to support.

For the time being, all new config options and code that is specific to a single vendor will be rejected.

I imagine that what could happen over time is for us to implement the methods needed for Headscale and OIDC to be an interface, where we can support GenericOIDC, GoogleOIDC, KeycloakOIDC, etc.

However, preferably, we will be quite conservative and keep it as small as possible. This is authentication code and bugs here have a lot larger implications than elsewhere.

The same goes for adding more options to the already overloaded config file.