Authentic 2 - Development #16854

OIDC Support of non-standard RP

13 juin 2017 12:31 - Mikaël Ates

Statut:	Rejeté	**Début:**	13 juin 2017
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:	OpenID Connect	**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

We may support RP that does not support JWT signature (though http://openid.net/specs/openid-connect-core-1_0.html#ServerMTI makes it mandatory) or add user attributes in the ID token (authorized by http://openid.net/specs/openid-connect-core-1_0.html#IDToken).

Seems it's a common behavior of famous running OP.

That might make authentic support things like https://github.com/authomatic/authomatic/blob/master/authomatic/providers/oauth2.py.

Révisions associées

Révision 2de800e0 - 13 juin 2017 19:40 - Benjamin Dauvergne

idp_oidc: fill id_token also with user's info (#16854)

Historique

#1 - 13 juin 2017 12:32 - Mikaël Ates

- Tracker changé de Support à Development

#2 - 13 juin 2017 12:32 - Mikaël Ates

- Description mis à jour

#3 - 13 juin 2017 13:41 - Benjamin Dauvergne

- Fichier 0001-idp_oidc-fill-id_token-also-with-user-s-info-fixes-1.patch ajouté

- Patch proposed changé de Non à Oui

#4 - 13 juin 2017 13:41 - Benjamin Dauvergne

- Assigné à mis à Benjamin Dauvergne

#5 - 13 juin 2017 14:39 - Benjamin Dauvergne

Par contre les signatures on les enlève jamais, si un RP ne veut pas le vérifier c'est son problème, il n'y a jamais de souci d'incompatibilité à ce niveau à ma connaissance (voir France Connect qui signe mais où tout le monde ignore la signature, la clé publique n'étant même pas diffusée).

#6 - 13 juin 2017 14:48 - Frédéric Péters

Ce que d'autres services font (et ce que le module plone assume, qui n'a rien pour désérialiser l'idtoken), c'est qu'il y ait des infos de l'usager directement dans le json de retour du oidc-token.

De manière moche sur ton patch, ça ressemblerait à ça :

```
--- a/src/authentic2_idp_oidc/views.py
+++ b/src/authentic2_idp_oidc/views.py
@@ -368,6 +368,7 @@ def token(request, *args, **kwargs):
         'access_token': unicode(access_token.uuid),
         'token_type': 'Bearer',
         'expires_in': expires_in,
+        'user_info': utils.create_user_info(client, oidc_code.user, oidc_code.scope_set()),
         'id_token': utils.make_idtoken(client, id_token),
     }), content_type='application/json')
     response['Cache-Control'] = 'no-store'
```

#7 - 13 juin 2017 15:37 - Benjamin Dauvergne

À ma connaissance aucun IdP OIDC ne fait ça :/ C'est sensé être compatible avec quoi ce truc Plone ?

#8 - 13 juin 2017 15:58 - Frédéric Péters

cf https://github.com/authomatic/authomatic/blob/master/authomatic/providers/oauth2.py , j'ai peut-être une lecture erronée du déroulé (et on s'est peut-être planté en ajoutant des trucs authentic dedans).

Mais ce que ça donnait, c'était un appel sur l'url token puis création d'un usager avec certaines infos puis ensuite l'appel au endpoint userinfo. (ce n'est pas des tests sur mon laptop, je ne peux pas rejouer facilement).

#9 - 13 juin 2017 16:22 - Benjamin Dauvergne

Ok mais c'est des fournisseurs OAuth2 issues de sites sociaux pas des RPs OIDC, ensuite la quasi totalité d'entre eux ont un endpoint user_info, c'est assez standard comme pratique.

Je comprends pas bien si il faudrait utiliser authomatic pour ajouter du login social ou si il faudrait qu'authentic soit compatible avec un truc existant dans authomatic (les URLs étant en dur je ne vois pas trop l'utilité). Où si ça un rapport fort avec du code dans plone que je ne vois pas ici.

#10 - 13 juin 2017 16:25 - Benjamin Dauvergne

Concernant les OPs OIDC qui mettent de la donnée dans l'id token il y a google qui recopie tout aux deux endroits (id_token et user_info), mais pas du tout dans la réponse du token endpoint.

#11 - 13 juin 2017 21:39 - Frédéric Péters

> Je comprends pas bien si il faudrait utiliser authomatic pour ajouter du login social ou si il faudrait qu'authentic soit compatible avec un truc existant dans authomatic (les URLs étant en dur je ne vois pas trop l'utilité). Où si ça un rapport fort avec du code dans plone que je ne vois pas ici.

Le module pour faire du SSO avec oidc dans Plone se base sur authomatic, il y a un besoin de SSO de Plone vers Authentic. Le module authomatic est orienté "IdP sociaux", ok et ça a peut-être imposé des choix, mais pour ajouter le SSO dans Plone, ça serait super pratique d'avoir juste à hériter d'une classe et y mettre une ou deux méthodes (genre pour faire scopes.split(' ') vu que par défaut ça attend une séparation par virgule, genre), plutôt que, je ne sais pas, développer de quoi faire un SP matchant parfaitement ce que produit authentic aujourd'hui.

L'idée étant aussi derrière que ce n'est pas juste ce module particulier, qu'ailleurs aussi on rencontrera des modules qui "font" de l'oidc mais qui ont été fait pour le "social" , avec les mêmes raccourcis.

#12 - 13 juin 2017 22:27 - Benjamin Dauvergne

Frédéric Péters a écrit :

> Le module pour faire du SSO avec oidc dans Plone se base sur authomatic, il y a un besoin de SSO de Plone vers Authentic. Le module authomatic est orienté "IdP sociaux", ok et ça a peut-être imposé des choix, mais pour ajouter le SSO dans Plone, ça serait super pratique d'avoir juste à hériter d'une classe et y mettre une ou deux

"peut-être imposé des choix" est certainement un euphémisme :)

> méthodes (genre pour faire scopes.split(' ') vu que par défaut ça attend une séparation par virgule, genre), plutôt que, je ne sais pas, développer de quoi faire un SP matchant parfaitement ce que produit authentic aujourd'hui.

En oauth2 le séparateur est toujours l'espace[1] (mais c'est peut être un exemple inventé).

> L'idée étant aussi derrière que ce n'est pas juste ce module particulier, qu'ailleurs aussi on rencontrera des modules qui "font" de l'oidc mais qui ont été fait pour le "social" , avec les mêmes raccourcis.

authomatic ne fait pas d'oidc, et ce qu'ils appellent oauth2 date souvent d'avant la standardisation, c'est donc du vieux code qui fait du vieil oauth2 car la plupart des réseaux sociaux n'ont pas fait le saut, les seuls à vraiment faire de l'OIDC sont google et microsoft il me semble. Néanmoins je pense que la classe Google d'authomatic devrait fonctionner vers authentic en adaptant les URIs et l'extracteur d'attributs, c'est globalement compatible avec ce qu'ils ont fait plus tard pour OIDC[2] (le endpoint user_info est le même par exemple).

[1] Et donc ce genre de code montre que authomatische ne vise pas l'OAuth2 des RFC mais le gloubiboulga qui le précéda:

```
    def _x_scope_parser(self, scope):
        """
        Google has space-separated scopes.
        """
        return ' '.join(scope)
```

[2] URL dans authomatic:

```
    user_authorization_url = 'https://accounts.google.com/o/oauth2/auth'
    access_token_url = 'https://accounts.google.com/o/oauth2/token'
    user_info_url = 'https://www.googleapis.com/oauth2/v3/userinfo?alt=json'
```

URL publiées dans les métadonnées OIDC de Google:

```
"issuer": "https://accounts.google.com",
"authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
"token_endpoint": "https://www.googleapis.com/oauth2/v4/token",
"userinfo_endpoint": "https://www.googleapis.com/oauth2/v3/userinfo",
"revocation_endpoint": "https://accounts.google.com/o/oauth2/revoke",
"jwks_uri": "https://www.googleapis.com/oauth2/v3/certs"
```

#13 - 13 juin 2017 22:32 - Frédéric Péters

Mais donc on n'a pas l'objectif de compatibilité facile avec les modules "oauth2" qu'on trouvera communément dans les logiciels tiers, ok.

#14 - 13 juin 2017 22:49 - Benjamin Dauvergne

Ça m'irait si ça allait dans un module authentic2_idp_whatever plutôt que authentic2_idp_oidc; ensuite tant que ça reste google ça semble matcher c'est déjà un bon début. Le problème que j'ai avec cet objectif c'est que ça ne marche pas, les logiciels qui visent twitter/facebook/etc... ont les URLs en dur dans leur code, il ne suffirait pas de mimer ces services pour que cela marche out of the box.

#15 - 13 juin 2017 22:55 - Frédéric Péters

> il ne suffirait pas de mimer ces services pour que cela marche out of the box.

Sûr, c'est juste que face à un module "oauth2/social" existant, l'adaptation pour fonctionner avec authentic pourrait être mineure (objectif "support of non-standard RP") plutôt que galère (situation actuelle face à des modules qui n'ont rien pour la manipulation de json web token).

Mais d'accord pour refuser l'objectif dans le cadre de authentic2_idp_oidc.

#16 - 19 juin 2017 14:48 - Benjamin Dauvergne

Ça m'irait aussi qu'on me donne ici précisément les différences entre ce que fait actuellement authentic et ce qu'il faudrait pour supporter authomatische, j'ai déjà dit qu'avoir user_info sur une réponse au endpoint /token était certainement une incompréhension de la part du lecteur du code, personne n'a jamais fait ça (et le code d'automatische me laisse à penser qu'il est tout à fait prévu pour interroger un endpoint user_info). Je préfèrerai que le module visé soit celui de Google (juste une énième fois l'impression que c'est le plus proche de nous).

Avec ces informations je pourrai donner une estimation de faisabilité plutôt que de botter en touche.

#17 - 19 juin 2017 14:55 - Frédéric Péters

J'étais parti sur un refus et donc qu'imio implémente sa propre classe dérivée d'authomatic qui fasse les choses à base de jwt etc.; c'est donc l'option qui a été prise le second jour de participation au sprint.

Reste sur le fond, lors de travaux d'assistance à connexion de RP, qu'on pourra rencontrer des RP n'implémentant pas la dernière norme correctement, et qu'il pourrait être plus facile d'adapter authentic que de faire évoluer ces modules, mais pour le cas précis d'authomatic/imio, l'option prise a donc été de le rendre compatible.

#18 - 20 juin 2017 15:50 - Benjamin Dauvergne

Je ne comprends pas cette insistance sur jwt qui est totalement inutile pour interopérer avec authentic2_idp_oidc (pour être clair, technocarte ne le valide même pas par exemple, ils ne regardent même pas dedans je pense) en fait, et donc je pense que la discussion ici n'a pas fonctionné. Le plus simple c'est authorize -> code -> token?code= -> acess_token -> user_info avec bearer et on obtient le sub (et les données sur l'utilisateur). Il n'y a pas de JWT à traiter là dedans.

Avec JWT on gagne juste un appel HTTP.

#19 - 20 juin 2017 16:21 - Frédéric Péters

Dans l'archi d'authomatic il y a nécessité d'avoir une clé primaire avant l'appel à user_info.

#20 - 20 juin 2017 17:15 - Benjamin Dauvergne

- Fichier 0001-implement-authentic-provider.patch ajouté

Ça m'étonne beaucoup car les utilisateurs sont créés dans BaseProvider._update_or_create_user() qui est appelé par

AuthorizationProvider.update_user() qui lui même appelle _access_user_info() juste avant[1] dont le boulot est justement d'appelé le endpoint user_info. Comme google ne sert pas de sub sur son endpoint access_token (et que authomatische n'utilise pas le JWT servi par google), je continue à penser qu'il y a une incompréhension dans le fonctionnement d'authomatische mais je veux bien voir le code produit pour voir ce qui a été fait.

De même dans les exemples (ici Django) on voit qu'il faut appeler user_info si on a pas d'user.id après le premier échange OAuth2:

```
def login(request, provider_name):

    # We we need the response object for the adapter.
    response = HttpResponse()

    # Start the login procedure.
    result = authomatic.login(DjangoAdapter(request, response), provider_name)

    # If there is no result, the login procedure is still pending.
    # Don't write anything to the response if there is no result!
    if result:
        # If there is result, the login procedure is over and we can write to
        # response.
        response.write('<a href="..">Home</a>')

        if result.error:
            # Login procedure finished with an error.
            response.write(
                '<h2>Damn that error: {0}</h2>'.format(result.error.message))

        elif result.user:
            # Hooray, we have the user!

            # OAuth 2.0 and OAuth 1.0a provide only limited user data on login,
            # We need to update the user to get more info.
            if not (result.user.name and result.user.id):
                result.user.update()
```

result.user.update() appelle update_user que je cite plus haut.

J'attache ce qui me semble une implémentation minimale (modulo les URLs).

[1]

```
    def update_user(self):
        """
        Updates the :attr:`.BaseProvider.user`.

        .. warning::
            Fetches the :attr:`.user_info_url`!

        :returns:
            :class:`.UserInfoResponse`

        """
        if self.user_info_url:
            response = self._access_user_info()
            self.user = self._update_or_create_user(response.data,
                                                    content=response.content)
            return authomatic.core.UserInfoResponse(self.user,
                                                    response.httplib_response)
```

#21 - 20 juin 2017 17:24 - Frédéric Péters

Benjamin on perd son temps dans ce ticket.

#22 - 21 juin 2017 11:44 - Benjamin Dauvergne

- Statut changé de Nouveau à Rejeté

Ok.

Fichiers

0001-idp_oidc-fill-id_token-also-with-user-s-info-fixes-1.patch	4,82 ko	13 juin 2017	Benjamin Dauvergne
0001-implement-authentic-provider.patch	2,16 ko	20 juin 2017	Benjamin Dauvergne