

jupyterhub / **oauthenticator** Public

<> **Code** Issues 46 Pull requests 7 Actions Projects Wiki Security

Commit f0c7002



minrk authored 2 weeks ago · ✖ 7/8 · Verified

Merge commit from fork
enforce email_verified in auth0

main · 17.4.0
2 parents [26d6bae](#) + [fda019c](#) commit [f0c7002](#)

2 files changed +78 -5 lines changed

Top

- oauthenticator
 - auth0.py
 - tests
 - test_auth0.py

2 files changed +78 -5 lines changed



oauthenticator/auth0.py



```

@@ -5,7 +5,8 @@
5 5  import os
6 6
7 7  from jupyterhub.auth import LocalAuthenticator
8 - from traitlets import Unicode, default
8 + from tornado.web import HTTPError
9 + from traitlets import Bool, Unicode, default
9 10
10 11  from .oauth2 import OAuthenticator
11 12

```

```

@@ -56,6 +57,15 @@ def _auth0_subdomain_default(self):
56 57         # This is allowed to be empty unless auth0_domain is not supplied
        either
57 58         return os.getenv("AUTH0_SUBDOMAIN", "")
58 59
60 +     allow_unverified_email = Bool(
61 +         False,
62 +         config=True,
63 +         help="""
64 +         Allow login with unverified email.
65 +         Not advisable, except for testing purposes.
66 +         """,
67 +     )
68 +
59 69     @default("logout_redirect_url")
60 70     def _logout_redirect_url_default(self):
61 71         return f"https://{self.auth0_domain}/v2/logout"
@@ -72,6 +82,31 @@ def _token_url_default(self):
72 82     def _userdata_url_default(self):
73 83         return f"https://{self.auth0_domain}/userinfo"
74 84
85 +     async def check_allowed(self, username, auth_model):
86 +         # A workaround for JupyterHub < 5.0 described in
87 +         # https://github.com/jupyterhub/oauthenticator/issues/621
88 +         if auth_model is None:
89 +             return True
90 +
91 +         # before considering allowing a username by being recognized in a list
92 +         # of usernames or similar, we must ensure that the authenticated user
93 +         # has a verified email and is part of hosted_domain if configured.
94 +         user_info = auth_model["auth_state"][self.user_auth_state_key]
95 +         user_email = user_info["email"]
96 +
97 +         if not user_info.get("email_verified"):
98 +             if self.allow_unverified_email:
99 +                 message = (
100 +                     f"Allowing login for {username} with unverified email
101 +                     {user_email}"
                )

```

```

102 +         self.log.warning(message)
103 +         else:
104 +             message = f"Login with unverified email {user_email} is not
            allowed"
105 +             self.log.warning(message)
106 +             raise HTTPError(403, message)
107 +
108 +             return await super().check_allowed(username, auth_model)
109 +

```

```

75 110         # _deprecated_oauth_aliases is used by deprecation logic in OAuthenticator
76 111         _deprecated_oauth_aliases = {
77 112             "username_key": ("username_claim", "16.0.0"),

```



oauthenticator/tests/test_auth0.py



```
@@ -24,13 +24,17 @@ def auth0_client(client):
```

```

24 24         return client
25 25
26 26
27 - def user_model():
27 + def user_model(extra=None):
28 28         """Return a user model"""
29 - return {
29 + model = {
30 30             "email": "user1@example.com",
31 31             "name": "user1",
32 32             "groups": ["group1"],
33 +             "email_verified": True,
33 34         }
35 + if extra:
36 +     model.update(extra)
37 + return model
34 38
35 39
36 40 @mark.parametrize(
@@ -113,7 +117,6 @@ async def test_auth0(
113 117     expect_allowed,
114 118     expect_admin,
115 119 ):

```



```

116 - print(f"Running test variation id {test_variation_id}")
117 120 c = Config()
118 121 c.Auth0Authenticator = Config(class_config)
119 122 c.Auth0Authenticator.auth0_domain = AUTH0_DOMAIN
@@ -138,7 +141,42 @@ async def test_auth0(
138 141     assert user_info == handled_user_model
139 142     assert auth_model["name"] == user_info[authenticator.username_claim]
140 143     else:
141 -     assert auth_model == None
144 +     assert auth_model is None
145 +
146 +
147 + @mark.parametrize(
148 +     "email_verified, allow_unverified, expect_allowed",
149 +     [
150 +         (True, False, True),
151 +         (False, False, False),
152 +         (False, True, True),
153 +     ],
154 + )
155 + async def test_email_verified(
156 +     auth0_client, email_verified, allow_unverified, expect_allowed
157 + ):
158 +     c = Config()
159 +     c.Auth0Authenticator = Config()
160 +     c.Auth0Authenticator.auth0_domain = AUTH0_DOMAIN
161 +     c.Auth0Authenticator.username_claim = "email"
162 +     c.Auth0Authenticator.allow_all = True
163 +     if allow_unverified:
164 +         c.Auth0Authenticator.allow_unverified_email = allow_unverified
165 +     authenticator = Auth0Authenticator(config=c)
166 +     assert authenticator.allow_unverified_email == allow_unverified
167 +
168 +     handled_user_model = user_model({"email_verified": email_verified})
169 +     handler = auth0_client.handler_for_user(handled_user_model)
170 +
171 +     auth_model = await authenticator.get_authenticated_user(handler, None)
172 +     assert auth_model is not None
173 +     if expect_allowed:

```

```
174 +         allowed = await authenticator.check_allowed(auth_model["name"],
175 +             auth_model)
176 +         assert allowed
177 +     else:
178 +         with raises(web.HTTPError) as exc_info:
179 +             await authenticator.check_allowed(auth_model["name"], auth_model)
180 +             assert exc_info.value.status_code == 403
181
182 async def test_custom_logout(monkeypatch):
```



Comments 0



Please [sign in](#) to comment.