МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Факультет информатики
Кафедра информационных систем и технологий

Дисциплина
«Операционные системы»

ОТЧЕТ

по дополнительному заданию

Разработка python-пакета для беспарольной аутентификации в JupyterHub.

Студент: Недугов В. Г.
Группа: 6202-020302D

Преподаватель: Востокин С.В.

Самара, 2023

ВВЕДЕНИЕ

В настоящий момент для JupyterHub существует шесть аутентификаторов (https://tljh.jupyter.org/en/latest/topic/authenticator-configuration.html):

1. OAuthenticator — Google, GitHub, CILogon, GitLab, Globus, Mediawiki, auth0, универсальное подключение OpenID (для KeyCloak и т. д.) и другие методы аутентификации на основе OAuth.
2. LDAPAuthenticator — LDAP и Active Directory.
3. DummyAuthenticator — любое имя пользователя, один общий пароль.
4. FirstUseAuthenticator — пользователи устанавливают свой пароль при первом входе в систему. Аутентификатор по умолчанию, используемый в TLJH.
5. TmpAuthenticator — открывает JupyterHub для всего мира, создает нового пользователя каждый раз, когда кто-то входит в систему.
6. NativeAuthenticator — разрешает пользователям регистрироваться, добавлять проверку безопасности пароля и блокировать пользователей после неудачных попыток входа в систему.

Основным из которых является NativeAuthenticator, поставляющийся по умолчанию вместе с JupyterHub. За основу написания пакета был взят TmpAuthenticator (https://github.com/jupyterhub/tmpauthenticator). Он работает по принципу создания сессии и выдаче cookie любому пользователю, перешедшему на хендлер обработчика регистрации или авторизации. Его основной недостаток заключается в отсутствии пользователей администраторов и каких-либо еще привилегированных пользователей и групп, т.к. TmpAuthenticator выдает права простого пользователя всем и сразу запускает для них notebook при авторизации. Также его недостатком является то, что он сохраняет все сессии пользователей за все время и не производит их очистку. Если же у пользователя не успели истечь cookie, то он без проблем авторизируется и ему будет выдан прошлый notebook, в котором он работал. Таким образов время жизни notebook-ов ограничено cookie-сессией и их можно чистить.

Для решения данной задачи был изучен пакет NativeAuthenticator, в него была дописана соответствующая логика TmpAuthenticator. Также были устранены перечисленные недостатки TmpAuthenticator.

В виду отсутствия возможности работы под Linux для запуска и тестирования написанного пакета в ОС Windows 10 был использован Docker. Соответствующий docker-compose для запуска можно найти по ссылке (https://github.com/jupyterhub/jupyterhub-deploy-docker).

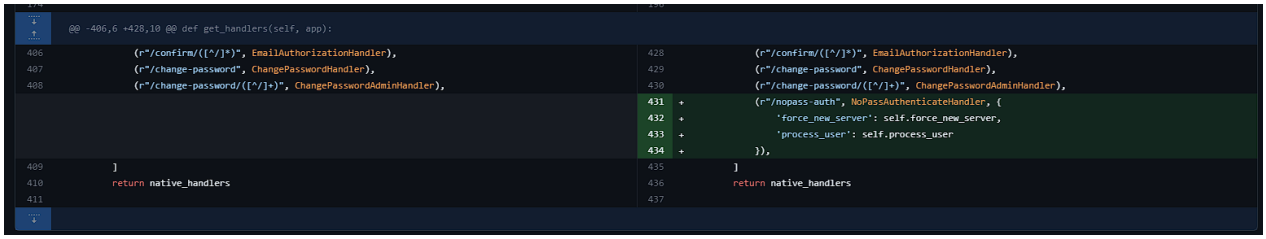
СБОРКА И ЗАПУСК DOCKER-КОНТЕЙНЕРА ДЛЯ ТЕСТИРОВАНИЯ

Сборка docker-контейнера для тестирования осуществлялась на основе файлов репозитория https://github.com/jupyterhub/jupyterhub-deploy-docker, небольшие изменения коснулись только Dockerfile-a, в котором 10-я строка была изменена на название и версию своего пакета. На данный момент стабильной работающий версией является пакет jupyterhub-tmpnativeauthenticator==1.0.8. Доступен по ссылке: https://pypi.org/project/jupyterhub-tmpnativeauthenticator/

В файле конфигов JupyterHub класс аутентификатора остался без изменения. В моем пакете и в оригинальном NativeAuthenticator названия классов одинаковы.

НАПИСАНИЕ СВОЕГО ПАКЕТА

Для работы был сделан форк репозитория NativeAuthenticator. Историю изменений и коммитов можно найти в моем репозитории по ссылке: https://github.com/Gorbacheb/tmpnativeauthenticator. NativeAuthenticator был расширен обработчиком из TmpAuthenticator:



```
@@ -406,6 +428,10 @@ def get_handlers(self, app):
            (r"/confirm/([^/]*)", EmailAuthorizationHandler),
            (r"/change-password", ChangePasswordHandler),
            (r"/change-password/([^/]+)", ChangePasswordAdminHandler),
+           (r"/nopass-auth", NoPassAuthenticateHandler, {
+               'force_new_server': self.force_new_server,
+               'process_user': self.process_user
+           }),
        ]
        return native_handlers
```

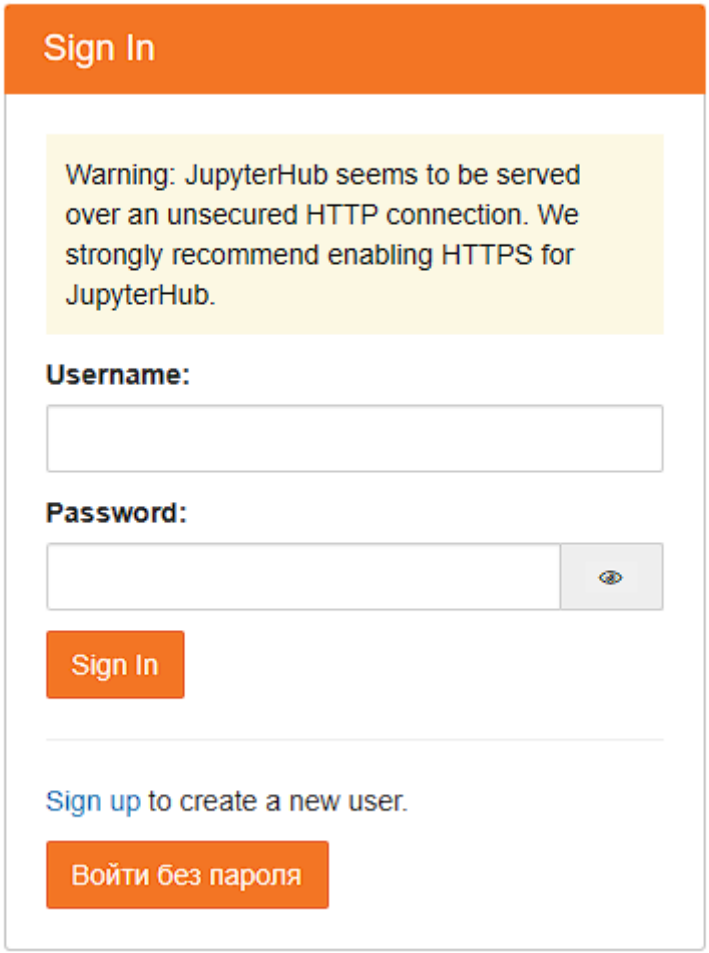
Были добавлены ссылки на страницах авторизации и регистрации JupyterHub для беспарольного входа:

```
nativeauthenticator/templates/native-login.html
@@ -77,6 +77,8 @@
            <a href="{{ base_url }}signup"> Sign up</a> to create a new user.
        </p>
        {% endif %}
+
+       <a href="{{ base_url }}nopass-auth" class="btn btn-jupyter"> Войти без пароля</a>
    </div>
</form>
</div>

nativeauthenticator/templates/signup.html
@@ -123,6 +123,9 @@
        <p>
            <a href="{{ base_url }}login">Login</a> with an existing user.
        </p>
+       <p>
+           <a href="{{ base_url }}nopass-auth">Temporary login</a> for a temporary login without password
+       </p>
    </div>
</form>
</div>
```

Вход осуществляется по адресу "/hub/nopass-auth".

Изменил страницу входа:



Для удаления неактивных пользователей с истекшим cookie при создании нового пользователя происходит поиск первого пользователя, у которого время последней активности отличается от текущего на значение tmp_user_lifetime в секундах. Данное значения устанавливается в конфигах хаба. Ссылка на коммит:

https://github.com/Gorbacheb/tmpnativeauthenticator/commit/e2fd9fe62fae2b4174f3f7bbc820b16f0b8ff2bf

Для создания пакета необходимо было арегистрироваться в PyPI. Для сборки и загрузки пакета в PyPI использовались команды:

python setup.py sdist

twine upload dist/*

Данный пакет (jupyterhub-tmpnativeauthenticator) доступен для скачивания по адресу https://pypi.org/project/jupyterhub-tmpnativeauthenticator/. Может быть установлен командой:

```
pip install jupyterhub-tmpnativeauthenticator
```