Бесшовный переход из мобильных приложений в Web

Оглавление

ПРЕДУПРЕЖДАЕМ

Функциональность находится в опытной эксплуатации. Возможны изменения в протоколах интеграции и конфигурации систем.

Функциональность бесшовного перехода позволяет пользователю, будучи аутентифицированным в мобильном приложении, переходить в веб-системы (личный кабинет, интернет-банк) сразу также в аутентифицированном состоянии.

Примеры использования:

- Часть функциональности реализована только в вебе. В мобильном приложении размещается только ссылка

Устройство

Предусловия

1. Аутентификация в мобильное приложение осуществляется при помощи RooX UIDM. Способ входа не имеет значения.
2. Аутентификация в веб-систему осуществляется при помощи RooX UIDM.
3. В настройках `sso-server` данному мобильному приложению разрешено выполнять переходы.
4. В настройках `sso-server` целевые адреса перехода зарегистрированы в качестве `redirect_uri`.

Сценарий работы

1. Пользователь выполняет аутентификацию в мобильном приложении.
2. `sso-server` в результате аутентификации выдает токен доступа. Мобильное приложение сохраняет токен.
3. Пользователь активирует ссылку перехода в веб. Это должна быть не просто ссылка, а визуальный элемент с навешенным обработчиком клика.
4. Мобильное приложение выполняет запрос к `sso-server` на инициирование новой сессии аутентификации для перехода в веб.
5. Мобильное приложение выполняет запрос к `sso-server` на получение OAuth-кода для перехода в веб.

```
POST https://{sso_base_url}/sso/oauth2/access_token
Accept: application/json
Content-Type: application/x-www-form-urlencoded
```

```
client_id={client_id}&
client_secret={client_secret}&
realm=/customer&
grant_type=urn:roox:params:oauth:grant-type:m2m-authorization-code&
service=dispatcher&
accessToken={access_token}
```

- client_id — идентификатор мобильного приложения
- client_secret — секрет, в данном сценарии — пустая строка
- access_token — токен доступа, полученный ранее при аутентификации

> ЗАМЕТКА
>
> Значение параметра service будет изменено на `service=native2web`.

6. `sso-server` создает OAuth код и возвращает в ответе:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
{
  "access_token": "46ccb2d9-fc46-49d2-9d2c-05d5e3aa62d1",
  "expires_in": 59
}
```

В ответе в поле `access_token` находится oauth code.

> ЗАМЕТКА
>
> Имя поля будет изменено на "code". Следует писать код парсинга как
> `response.contains("code")?response.get("code"):response.get("access_token")`

7. Мобильное приложение формирует ссылку особым образом: https://{consumer_endpoint_url}?code={oauth_code}&goto={target_url}

   Пример

```
https://ib.example.com/oauth2-consumer?
code=12312321&goto=https://ib.example.com/offers/1
```

8. Мобильное приложение открывает браузер по-умолчанию используя средства мобильной ОС.
9. Браузер открывается и переходит по ссылке.
10. Веб-приложение (или служебный компонент OAuth2 Consumer) получает oauth-код из ссылки и выполняет запрос к `sso-server` на обмен кода на токены. Эта часть сценария работает как обычно, как будто пользователь только что ввел логин-пароль в форме логина.

11. `sso-server` создает access token, refresh token и возвращает в ответе.
12. Веб-приложение (или OAuth2 Consumer) устанавливает токены в cookie и выполняет перенаправление на целевую страницу.

Компоненты, необходимые для работы

- `sso-server` (Основной сервис аутентификации: выполняет аутентификацию, предоставляет API для аутентификации, самообслуживания и администрирования)
- опционально: `oauth2-consumer-server` (Обработчик протокола OAuth2 от лица Service Provider)