Mlk Password Restore 2

Оглавление

ЗАМЕТКА

Рекомендуется использовать другой, более универсальный механизм восстановления пароля, описанный в API восстановления пароля

API предназначено для восстановления забытого пользователем пароля.

В данном варианте пользователь получает на адрес электронной почты email, содержащий ссылку для безопасного запуска сценария смены пароля. При смене пароля требуется подтверждение операции с помощью ОТП-кода, отправленного на номер телефона.

Для смены пароля системой через доверенное межсерверное обращение - используйте Provisioning API.

Сценарий восстановления пароля через SMS и/или EMAIL

Предусловия

- Пользователь не может войти в личный кабинет, поскольку забыл свой пароль
- Пользователю разрешено сменять пароль через EMAIL и RooX UIDM настроен соответствующе
- В профиле пользователя указан действующий телефон и электронная почта
- Пользователь открыл веб-браузер или мобильное приложение (далее называется как "Приложение")

Сценарий

1. Пользователь открывает форму входа и кликает по элементу "Восстановить пароль"
   a. Приложение отправляет запрос на старт сценария восстановления и получение execution
   b. Сервер отвечает описанием формы восстановления.

c. Приложение отображает форму идентификации

2. Пользователь вводит логин
   a. Приложение валидирует ввод
   b. Приложение отправляет запрос на идентификацию
   c. Сервер по идентификатору находит информацию о пользователе.
3. Сервер отвечает описанием формы ввода цифр номера телефона
   a. Приложение отображает форму ввода цифр номера телефона.
   b. Пользователь вводит часть цифр номера телефона
   c. Приложение валидирует ввод
   d. Приложение отправляет запрос на проверку цифр номера
   e. Сервер проверяет, что введенные пользователем цифры совпадают с последними цифрамим номера пользователя с логином, введенным на предыдущем шаге.
4. Сервер формирует ссылку для восстановления пароля
   a. Сервер сохраняет информацию о пользователе в БД и формирует уникальную ссылку на форму смены пароля для данного пользователя.
   b. Сервер отправляет письмо со сгенерированной ссылкой на email, указанный в профиле пользователя с данным логином.
   c. Сервер отвечает сообщением об успешной отправке email
5. Пользователь открывает ссылку из письма в браузере
   a. Сервер по коду из ссылки ищет информацию о запущенном сценарии восстановления пароля.
   b. Сервер в случае успешного поиска отвечает описанием формы старта сценария смены пароля
   c. Приложение отображает форму для начала сценария смены пароля и отображает часть номера телефона, на который будет отправлен ОТП-код.
6. Пользователь продолжает сценарий
   a. Приложение отправляет запрос на выпуск одноразового пароля
   b. Сервер генерирует второй одноразовый код и отправляет его на телефон, указанный в профиле
   c. Сервер отвечает описанием формы ввода одноразового кода с методом `SMS`
   d. Приложение отображает форму ввода второго одноразового кода
7. Пользователь вводит одноразовый код, полученный в SMS
   a. Приложение валидирует ввод
   b. Приложение отправляет запрос на проверку одноразового кода
   c. Сервер проверяет код
      i. Если код введен неверно, сервер отвечает ошибкой и, возможно, предоставляет еще попытку ввода
      ii. Если код введен верно, сервер отвечает описанием формы ввода нового пароля
      iii. Приложение отображает форму ввода нового пароля или форму повторного ввода первого кода, в зависимости от ответа сервера
8. Пользователь вводит новый пароль
   a. Приложение валидирует ввод

b. Приложение отправляет запрос на установку нового пароля

c. Сервер проверяет пароль согласно парольным политикам из конфигурации RooX UIDM

d. Сервер устанавливает новый пароль в БД RooX UIDM (только при использовании собственной БД RooX UIDM)

e. Сервер устанавливает новый пароль во внешнем хранилище учетных записей (только при использовании внешнего хранилища учетных записей)

f. Сервер записывает событие в БД Аудита `sso.credentials_change.success`

g. Сервер подтверждает смену пароля

h. Приложение обрабатывает ответ и перенаправляет пользователя на необходимую страницу

Постусловия

- Установлен новый пароль в БД RooX UIDM, таблица Credentials (только при использовании собственной БД RooX UIDM)
- Установлен новый пароль во внешнем хранилище учетных записей (только при использовании внешнего хранилища учетных записей)
- В БД Аудита запротоколировано событие `sso.credentials_change.success`

Замечания по работе сценария

1. Шаг 2 (проверка цифр номера телефона пользователя) может быть пропущен. Необходимость выполнения шага определяется значением `true` параметра `com.rooxteam.sso.restore_password.phone_verification.enabled` в настройках сервера RooX UIDM.
2. Все запросы должны быть выполнены в приведенной последовательности, так как параметр `execution` из каждого ответа используется как параметр в последующих запросах.
3. Ссылка на форму смены пароля, которая отправляется по email, технически является одноразовым кодом доступа (OTP) категории `restore-password`. URL содержит в себе параметр code, в котором зашифрована информация о сессии, к которому привязан данный код. Таким образом, для ссылки применимы все особенности механизмов OTP-кодов RooX UIDM: ограничение на время действия ссылки, ограничение на число одновременно выпущенных ссылок. Информация о выпущенных ссылках, как и для обычных OTP, хранится в таблице `OTPCode` в зашифрованном виде, пока ссылка является валидной и расшифровывается по окончании срока действия ссылки для возможности клиентского обслуживания.

ВАЖНО

Возвращаемое значение <execution> в каждом из запросов к RooX UIDM может обновляться. Необходимо использовать самое актуальное значение.

Старт сценария

Для начала сценария создания привязки необходимо отправить запрос в RooX UIDM на `/sso/auth/start-password-recovery`. В ответе будет содержаться параметр <execution>, который необходимо включить в следующий запрос к RooX UIDM.

Так же в ответе содержится описание формы ввода идентификатора пользователя, который восстанавливает пароль.

Формат запроса

```
GET /sso/auth/start-password-recovery?client_id=<client_id>

Host: <sso_host>
Accept: application/json
```

Параметры

- <sso_host> - базовый адрес сервера RooX UIDM, например sso.rooxteam.com
- <client_id> - идентификатор клиента, например selfcare

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы, а так же <execution>, который необходимо включить в следующий запрос к RooX UIDM.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```

```
{
  "execution": "<execution_value>",
  "form": {
    "name": "fieldForm",
    "fields": {
      "login": {
        "constraints": [
          {
            "name": "NotNull"
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<serverUrl>",
  "step": "enter_user_login"
}
```

Поля ответа

- <execution_value> - значение параметра <execution> для следующего запроса к RooX UIDM
- <form> - описание формы
- <serverUrl> - адрес сервера, на который необходимо отправить следующий запрос с клиента
- <step> - обозначение текущего шага сценария, в данном случае отображается форма поиска пользователя

Состав формы говорит, что следует отобразить форму ввода идентификатора пользователя и передать введенное значение в следующем запросе к API. Имя поля `name`, ограничение - значение не должно быть пустым.

Идентификация

UI отображает поле ввода идентификатора

Пользователь вводит свой идентификатор, после чего приложение выполняет запрос на идентификацию.

Формат запроса

```
POST /sso/auth/_get-user-email

Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded

execution=<execution>&
login=<login>
_eventId=send
```

Параметры запроса

- login - веденный логин пользователя
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `send`
- execution - значение равно значению поля ‹execution› полученному из ответа на предыдущий запрос к API

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы ввода OTP, а так же ‹execution›, который необходимо включить в следующий запрос к RooX UIDM.

В зависимости от настроек RooX UIDM OTP отправляется по электронной почте или по SMS. Поле 'method' укажет на использованный способ.

ЗАМЕТКА

В целях защиты персональных данных если пользователь на первом шаге ввел несуществующий логин, email или номер телефона, сервер не говорит, что пользователь не найден

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
```

```
{
  "execution": "<execution_value>",
  "form": {
    "name": "fieldForm",
    "fields": {
      "msisdn": {
        "constraints": [
          {
            "name": "NotNull"
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<serverUrl>",
  "step": "enter_phone_digits"
}
```

Поля ответа

- <execution_value> - значение параметра <execution> для следующего запроса к RooX UIDM
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода цифр номера
- form - описание формы
- <serverUrl> - адрес сервера, на который необходимо отправить следующий запрос с клиента

Состав формы говорит, что следует отобразить форму ввода цифр и передать введенное значение в следующем запросе к API. Имя поля - `name`, ограничение - значение не должно быть пустым.

Проверка цифр номера

После ввода идентификатора пользователь вводит последние несколько цифр номера своего телефона. Сервер проверяет корректность ввода.

```
POST /sso/auth/_analyze-phone-digits
Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded

execution=<execution>&
msisdn=<msisdnDigits>&
_eventId=send
```

Параметры запроса

- execution - значение равно значению поля <execution> полученному из ответа на предыдущий запрос к API
- msisdn - заданное количество последних цифр номера телефона.
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `validate`

Формат успешного ответа в случае, когда требуется ввод второго OTP кода, отправленного другим способом

В ответе содержится JSON объект, содержащий описание формы ввода OTP, а так же <execution>, который необходимо включить в следующий запрос к RooX UIDM.

В зависимости от настроек RooX UIDM второй OTP код отправляется по электронной почте или по SMS. Поле 'method' укажет на используемый способ.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
```

```
{
  "execution": "<execution_value>",
  "serverUrl": "<serverUrl>",
  "step": "email_sent"
}
```

Поля ответа

- <execution_value> - значение параметра <execution> для следующего запроса к RooX UIDM
- <serverUrl> - игнорируется
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода OTP кода

Открытие ссылки из email и старт сценария смены пароля

После открытия ссылки, полученной по email начинается сценарий смены пароля для данного пользователя.

```
GET /sso/auth/login-password-change?code=<code>&client_id=<client_id>
Host: <sso_host>
Accept: application/json
```

Параметры запроса

- <code> - закодированная информация о сценарии восстановления пароля
- <sso_host> - базовый адрес сервера RooX UIDM, например sso.rooxteam.com
- <client_id> - идентификатор клиента, например selfcare

ЗАМЕТКА

данный URL формируется самим сервером RooX UIDM и отправляется клиенту внутри email-сообщения. Формировать его на стороне приложения не требуется.

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы смены пароля, а так же <execution>, который необходимо включить в следующий запрос к RooX UIDM.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```

```
{
  "execution": "<execution_value>",
  "view": {
    "msisdn": "<msisdnDigits>"
  },
  "serverUrl": "<serverUrl>",
  "step": "render"
}
```

Поля ответа

- <execution_value> - значение параметра <execution> для следующего запроса к RooX UIDM
- <msisdnDigits> - несколько последних цифр номера, на которые будет отправлен ОТП-код, для отображения пользователю.

- step - обозначение текущего шага сценария, в данном случае отображается форма для старта сценария смены пароля.
- <serverUrl> - адрес сервера, на который необходимо отправить следующий запрос с клиента

Запрос ОТП-кода

Пользователь подтверждает, что необходимо отправить ОТП код на номер, заканчивающийся на полученные в предыдущем шаге цифры.

```
POST /sso/auth/login-password-change
Accept: application/json
Content-Type: application/x-www-form-urlencoded

execution=<execution>&
_eventId=proceed
```

Параметры запроса

- execution - значение равно значению поля <execution> полученному из ответа на предыдущий запрос к API
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `proceed`

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы ввода OTP, а так же <execution>, который необходимо включить в следующий запрос к RooX UIDM.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
```

```
{
  "execution": "<execution_value>",
  "view": {
    "nextOtpCodePeriod": 9,
    "otpCodeAvailableAttempts": 6,
    "method": "SMS",
    "expireOtpCodeTime": 21599,
    "blockedFor": 0,
    "isBlocked": false,
    "otpCodeNumber": 4,
    "msisdn": "6549",
    "nextOtpPeriod": 9
  },
  "form": {
    "name": "otpForm",
    "fields": {
      "otpCode": {
        "constraints": [
          {
            "name": "NotNull"
          },
```

```
          {
            "name": "Size",
            "attributes": {
              "min": 4,
              "max": 2147483647
            }
          },
          {
            "name": "Pattern",
            "attributes": {
              "flags": [],
              "regexp": "^[0-9]+$"
            }
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<serverUrl>",
  "step": "enter_otp_form"
}
```


Поля ответа

- <execution_value> - значение параметра <execution> для следующего запроса к RooX UIDM
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода OTP кода
- form - описание формы
- view - информация о состоянии сценария, используется для отображения на UI
- view.method* - как был отправлен текущий код, варианты: SMS, EMAIL
- view.nextOtpCodePeriod - время в секундах до наступления возможности отправки нового OTP кода
- view.otpCodeAvailableAttempts - количество оставшихся попыток ввода OTP кода
- view.isBlocked - признак временной блокировки пользователя, ввод OTP невозможен, следует заблокировать окно ввода
- view.blockedFor - если установлен isBlocked, то поле содержит время до разблокировки в секундах
- view.msisdn - номер телефона, на который будет отправлен OTP SMS в случае, если сейчас отрабатывает method=SMS
- view.email - адрес электронной почты, на который будет отправлен OTP EMAIL
- view.otpCodeNumber - порядковый номер сообщения, сбрасывается ежедневно в 0:00
- view.expireOtpCodeTime - время жизни OTP кода в секундах. По истечении времени код будет считаться невалидным

Состав формы говорит, что следует отобразить форму ввода OTP кода и передать введенное значение в следующем запросе к API. Имя поля `otpCode`, ограничение - значение не должно быть пустым, содержать ровно 4 символа и соответствовать регулярному выражению `^[0-9]+$` (в данном случае - содержать только цифры).

Ввод OTP кода

После ввода OTP кода приложение отправляет OTP на проверку.

```
POST /sso/auth/_otp-sms?otpCode=<code>&execution=<execution>&_eventId=validate
Host: <sso_host>
```

```
Accept: application/json
```

Параметры запроса

- otpCode - введенный OTP код
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `validate`
- execution - значение равно значению поля ‹execution› полученному из ответа на предыдущий запрос к API

В зависимости от правильности введенного OTP кода и настроек сервера, ответ сервера будет указывать на способ продолжения сценария:

- step=enter_otp_form и непустой form.errors - код введен неверно или произошла другая ошибка, остаемся на текущем этапе сценария
- step=enter_credentials - код введен верно и сервер не требует проверки второго OTP-кода; следует отобразить форму создания нового пароля

Формат успешного ответа, Пользователь ввел корректный ОТП-код.

В ответе содержится JSON объект, содержащий описание формы данных для смены пароля, а так же ‹execution›, который необходимо включить в следующий запрос к RooX UIDM.

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

```
{
  "execution": "<execution_value>",
  "view": {
    "username": "<username>"
  },
  "form": {
    "name": "credentialsForm",
    "fields": {},
    "errors": []
  },
  "serverUrl": "<serverUrl>",
  "step": "enter_credentials"
}
```

Поля ответа

- ‹execution_value› - значение параметра ‹execution› для следующего запроса к RooX UIDM
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода новых учетных данных
- form - описание формы
- view - информация о состоянии сценария, используется для отображения на UI
- view.username* - имя (логин) текущего пользователя, который выполняет смену пароля.

Состав формы говорит, что следует отобразить форму ввода учетных данных.

Установка нового пароля и завершение сценария

Формат запроса

```
POST /sso/auth/login-password-change

Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded

execution=<execution_value>&
newpassword=<newPassword>&
_eventId=send
```

Параметры запроса

- execution - значение равно значению поля ‹execution› полученному из ответа на предыдущий запрос к API
- newpassword - введенный новый пароль
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `send`

Формат успешного ответа

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
```

```
{
  "step": "redirect",
  "location": "/sso/auth/complete"
}
```

Поля ответа

- step - обозначение текущего шага сценария, в данном случае необходимо выполнить редирект при завершении сценария
- location - адрес, на который необходимо выполнить редирект

Общие для всех шагов сообщения об ошибках

Код ошибки	Описание причины возникновения
invalid_credentials	Пользователь ввел неправильные данные учетной записи и не был аутентифицирован.
expired_password	Время действия пароля, введенного пользователем, истекло. Пользователь не был аутентифицирован.
user_blocked	Учетная запись пользователя заблокирована.

ip_blocked	IP адресс, с которого пользователь обращается в RooX UIDM, заблокирован.
invalid_captcha	Введенная пользователем CAPTCHA неверная.
need_captcha	Для продолжения сценария пользователь должен ввести CAPTCHA.
error	Пользователь не был аутентифицирован в результате неожиданного ответа от внешней системы.
login-by-otp-disabled	Пользователь попытался аутентифицироваться через сценарий с аутентификацией по OTP, но это невозможно, поскольку данный функционал отключен конфигурационным ключем.
error_sending_otp	Невозможно отправить OTP пользователю.
too_many_sms	Невозможно отправить OTP пользователю, поскольку превышено число допустимых повторных запросов отправки OTP. Примечание: OTP может быть отправлен пользователю произвольным способом, а не только через sms сообщение.
too_many_wrong_code	Превышено число допустимых попыток ввода OTP.
invalid_otp	Введенный OTP неверный.
validate-otp-fail	Введенный OTP неверный.
otp_expired	Время действия введенного OTP истекло.
otp-expired	Время действия введенного OTP истекло.
otp-error	Сценарий ввода OTP завершился неуспешно.
external-api-error	Выполнение сценария невозможно, поскольку получен неожиданный ответ от внешней системы или внешняя система не доступна.
external-system-bad-response	Выполнение сценария невозможно, поскольку получен неожиданный ответ от внешней системы или внешняя система не доступна.
msisdn-is-not-b2b	Введенный msisdn не принадлежит реалму b2b.
too_many_attempts	Превышено число допустимых попыток изменения пароля.

user-is-not-allowed	Данному пользователю запрещено продолжать данный сценарий.
error_password_change	Сценарий изменения пароля не был успешно завершен. Пароль не был изменен.
no_email_found	Пользователь не найден или у пользователя не существует email.
msisdn-not-exists	Пользователя с заданным msisdn не существует.
principal-not-exists	Пользователь не существует.
no-principal-found	Пользователь не существует.
user-not-found	Пользователь не существует.
login-exists	Невозможно зарегестрировать пользователя с заданным login, поскольку пользователь с заданным login уже существует.
login_already_exists	Невозможно изменить login пользователя на заданный, поскольку пользователь с заданным login уже существует.
email-exists	Пользователь с заданным email уже существует.
user-exists	Невозможно зарегестрировать пользователя с заданными msisdn или email или login, поскольку пользователь с такими параметрами уже существует.
email_verification_failed	Невозможно подтвердить email.
reset_required	Пароль или не был задан либо был сброшен в результате каких-то действий, необходимо создать новый пароль. Отличается от expired_password, поскольку в том случае пароль существует и был провалидирован, но требуется смена. В данном случае пароль не провалидирован, поскольку не задан.

Примеры ответов об ошибке

Не указан или передан пустой параметр <execution>

Формат ответа с ошибкой

```
HTTP/1.1 400 Bad Request
```

```
{
  "error": "invalid_grant",
  "error_description": "The provided access grant is invalid, expired, or revoked."
}
```

Не указан параметр <_eventId>

Формат ответа аналогичен ответу на первом шаге: Старт сценария

Указан неверный параметр <_eventId>

Формат ответа аналогичен ответу на первом шаге: Старт сценария, с заполненным списком ошибок в теле JSON ответа:

```
{
  "form": {
    "errors": [
      {
        "field": "username",
        "message": "may not be null"
      },
      {
        "field": "password",
        "message": "may not be null"
      }
    ]
  }
}
```