Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

«Оценка погрешности на основании проведения прямых измерений»

по дисциплине

“Информационная безопасность»

Выполнил:

Студенты группы P3430,

Тарасов Иван Сергеевич,

Проверил:

Рыбаков Степан Дмитриевич

 Санкт-Петербург

2025

Ссылка на репозиторий: [ITMO-University-labs/infosecurity-lab1](https://github.com/ITMO-University-labs/infosecurity-lab1)

**Описание проекта**

Проект выполнен на языке Java с использованием фреймворка Spring Boot.

*Список эндпоинтов*:

1) POST /auth/registration – регистрация пользователя. Требует наличие тела запроса в виде:

{

    "email": "ivan@ya.ru",

    "nickname": "IvanTarasov",

    "password": "123456"

}

Возвращает токен аутентификации

{

    "token": “<access token>”

}

Возвращает код **401**, если пользователь с такой почтой уже существует.

2) POST /auth/login – авторизация пользователя. Требует наличие тела запроса в виде:

{

    "email": "ivan@ya.ru",

    "nickname": "IvanTarasov",

    "password": "123456"

}

Возвращает токен аутентификации

{

    "token": “<access token>”

}

Возвращает код **401**, если в теле указан неверный логин или пароль.

3) GET /auth/refreshToken – обновление токена аутентификации в случае истечения его срока жизни. Требует наличие заголовка “Authorization”: “Bearer <access token>”.

Возвращает обновлённый токен аутентификации

{

    "token": “<access token>”

}

Возвращает код **401**, если истек срок жизни refresh токена или в случае повреждения информации о почте в access токене.

4) GET /api/data – возвращает список всех пользователей. Требуется авторизация!

Тело ответа:

[

    {

        "email": "ivan@ya.ru",

        "nickname": "IvanTarasov"

    },

{

        "email": "user@example.ru",

        "nickname": "UserName"

    },

…

]

5) DELETE /api/delete-user-by-email/{email} – удаляет пользователя по его почте. Требуется авторизация! Требуется указать в url адрес почты, например, [**/api/delete-user-by-email/user@example.ru**](mailto:/api/delete-user-by-email/user@example.ru). Данный эндпоинт ничего не возвращает в теле ответа.

**Реализация защит**

1. *SQL-инъекции*

Защита от SQL-инъекций происходит благодаря фреймворку Spring Data JPA, который под капотом использует Hibernate. Если бы на backend использовались голые запросы и параметры передавались бы строковой конкатенацией, например, **select \* from users where** **email= + ’test@example.com’** то можно было бы выполнить уязвимый запрос, который возвращал бы всех пользователей, даже несмотря на то, что почты [**test@example.com**](mailto:test@example.com)не существует, например вот так

**select \* from users where** **email=’test@example.com’ or 1=1;**

Условие **1=1** всегда истина.

Hibernate же не использует строковую конкатенацию. Параметры передаются, как bind-параметры. То есть фактически выполняется **select \* from users where email = ?**. Значение email подставляется отдельным параметром, а не как часть строки SQL.

2. *XSS-атака*

Так как я использую RestController’ы, соответственно, эти контроллеры возвращают данные в виде JSON, а не HTML. Даже если поле содержит **<script>alert(1)</script>**, оно будет сериализовано так:

{

"name": "<script>alert(1)</script>"

}

Браузер, получив JSON по Content-Type: application/json, не будет интерпретировать содержимое как HTML/JS, а просто как данные.  
Чтобы XSS случился, нужно было бы отрендерить эти данные в HTML без экранирования (например, вставить их в innerHTML).