

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

По дисциплине «Информационная безопасность»
Лабораторная работа №1
Разработка защищенного REST API с интеграцией в CI/CD

Студент:

Дениченко Александр Олегович Р3412

Практик:

Маркина Татьяна Анатольевна

Санкт-Петербург
2025 г.

Цель

Получить практический опыт разработки безопасного backend-приложения с автоматизированной проверкой кода на уязвимости. Освоить принципы защиты от OWASP Top 10 и интеграцию инструментов безопасности в процесс разработки.

Выбранные технологии

Java/Spring Boot, Spring Data JPA (Hibernate), H2 (in-memory), JWT (HS256), BCrypt, SAST SpotBugs, SCA Snyk, maven.

Выполнение

Проект был создан в GitHub (<https://github.com/Alex-de-bug/security-lab-1>). Реализовано 3 метода, где есть следующие эндпоинты:

- POST /auth/login — метод для аутентификации пользователя.
- POST /auth/register — метод для регистрации пользователя.
- GET /api/data — возвращает список всех постов. Доступ только у аутентифицированных пользователей.
- POST /api/posts - создает новый пост от имени текущего пользователя. Доступ только у аутентифицированных пользователей.
- GET /api/posts/my - возвращает все посты текущего пользователя. Доступ только у аутентифицированных пользователей.
- Swagger UI: /swagger-ui.html
- H2 Console: /h2-console

Меры защиты

Аутентификация и сессии. Stateless JWT Bearer (HS256), срок жизни из 86400000, секрет был сгенерирован на сайте для генерации JWT. Пароли — только в виде хеша BCrypt. Все запросы, кроме /auth/**, /swagger-ui/**, /v3/api-docs/** и /h2-console/**, требуют аутентификации.

```
.authorizeHttpRequests(  
    auth -> auth.requestMatchers("/auth/**")  
        .permitAll()  
        .requestMatchers("/h2-console/**")  
        .permitAll()  
        .requestMatchers("/swagger-ui/**", "/v3/api-docs/**", "/swagger-ui.html")  
        .permitAll()  
        .anyRequest()  
        .authenticated());
```

SQLi. Доступ к данным через Spring Data JPA и параметризованные запросы; отсутствует ручная конкатенация SQL.

XSS. Текстовые поля экранируются OWASP Java Encoder (утилита Sanitizer).

SAST

Первый запуск в файле ../target/spotbugsXml(0).xml. Обнаруженные проблемы:

В классе ApiController (ApiController.java, строки 39–180):

- Конструктор ApiController(PostService, UserRepository) сохраняет ссылку на изменяемый объект PostService в поле postService.
- Это может привести к тому, что внешний код, владеющий этим объектом, сможет непреднамеренно изменить внутреннее состояние контроллера.

В классе Post (Post.java, строки 14–47):

- Метод getAuthor() возвращает внутренний объект author напрямую, позволяя внешнему коду изменить его.
- Конструктор Post(String, String, User) сохраняет ссылку на внешний объект User в поле author.
- Метод setAuthor(User) устанавливает напрямую ссылку на изменяемый объект User.

Исправления, которые нужно было сделать

Класс Post:

- Добавлен безопасный метод getAuthorInfo(), возвращающий неизменяемый объект UserInfo с копией данных автора
- Добавлена валидация параметров с @NotNull и Objects.requireNonNull()

Класс ApiController:

- Добавлена валидация входных параметров в конструкторе
- Заменены все вызовы post.getAuthor().getUsername() на безопасный post.getAuthorInfo().getUsername()

Итог содержится в файле ../target/spotbugsXml(1).xml.

SCA

Для dependency check был выбран snyk. Была сразу поправлены все версии библиотек на новые и вот итог:

```
93     "packageManager": "maven",
94     "ignoreSettings": {
95       "adminOnly": false,
96       "reasonRequired": false,
97       "disregardFilesystemIgnores": false
98     },
99     "summary": "No known vulnerabilities",
100    "filesystemPolicy": false,
101    "uniqueCount": 0,
102    "projectName": "com.example:securityapi",
103    "displayTargetFile": "pom.xml",
104    "hasUnknownVersions": false,
105    "path": "/home/runner/work/security-lab-1/security-lab-1/securityapi"
106  }
107
```

Файл есть в ../target/ под названием snyk-report.json

Тестирование приложения

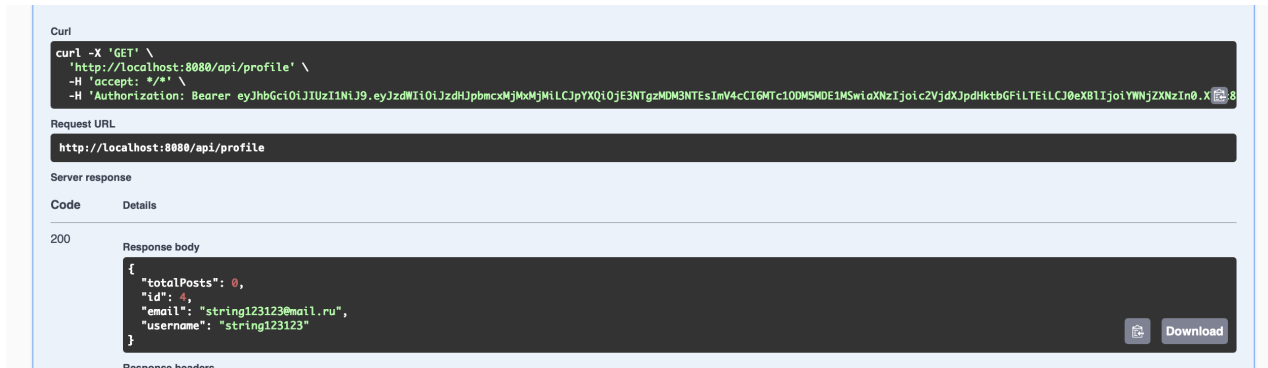
Запрос на регистрацию



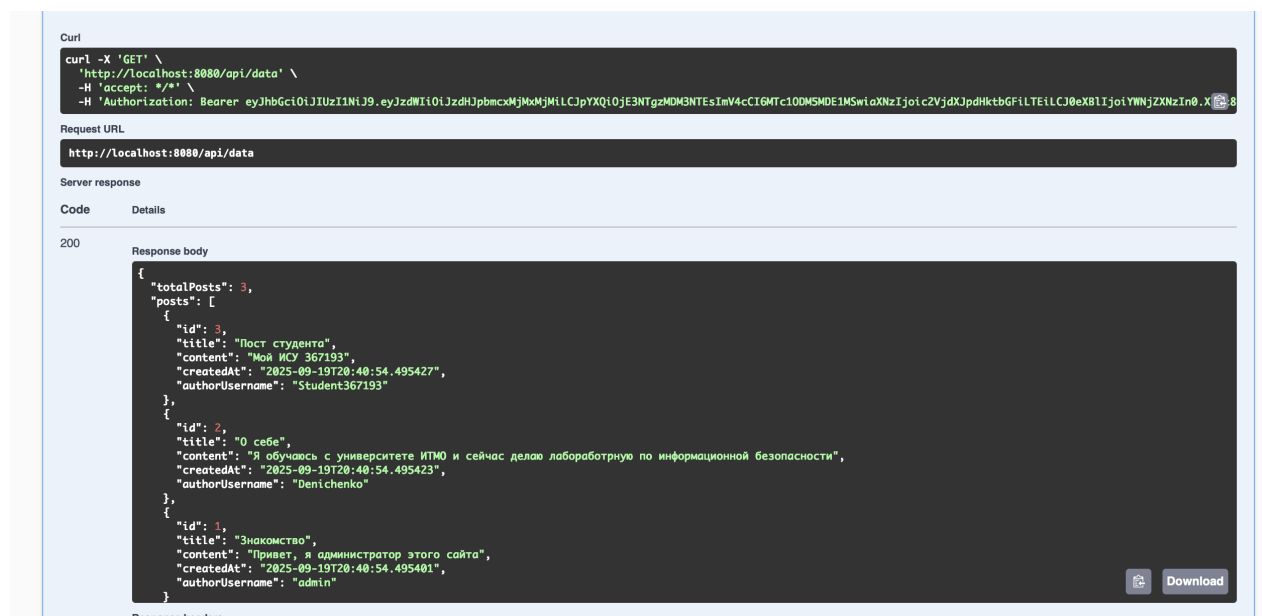
Запрос на логин



Запрос на информацию о профиле



Запрос на все посты



Результаты

Реализован защищённый REST API: аутентификация/регистрация, операции с постами, защита от XSS/SQLi, stateless JWT. Подготовлены и приложены отчёты SCA; настроена Swagger-документация. Последний верный CI (<https://github.com/Alex-de-bug/security-lab-1/actions/runs/17864752066>)