МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Базы данных»
Тема: Тестирование БД на безопасность

Студент гр. 3341 _____ Шаповаленко Е.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2025

Цель работы.

Целью данной лабораторной работы является изучение уязвимостей БД к SQL-инъекциям.

Задание.

Сделать простой web-сервер для выполнения запросов из ЛР3, например с express.js. Не обязательно делать авторизацию и т.п., хватит одного эдпоинта на каждый запрос, с параметрами запроса как query parameters.

Намеренно сделайте несколько (2-3) запроса, подверженных SQL-инъекциям

Проверьте Ваше API с помощью sqlmap (или чего-то аналогичного), передав эндпоинты в качестве целей атаки. Посмотрите, какие уязвимости он нашёл (и не нашёл), опишите пути к исправлению.

+2 балла, если напишете эндпоинт с уязвимостью, которая не находится sqlmap-ом.

Вариант 21

Пусть требуется создать программную систему, предназначенную для продавца журналов/комиксов. Такая система должна обеспечивать хранение сведений об имеющихся в магазине журналах, о читателях журналов и списке магазинов. Для каждого журнала в БД должны храниться следующие сведения: название журнала, серия, автор (ы), издательство, год издания, число экземпляров в каждом магазине, а также ISBN и дата продажи журнала. Сведения о читателях библиотеки должны включать почту (email), ФИО покупателя, дату рождения, адрес, номер телефон. Нужно учесть, что покупатели могут делать заказы с разных магазинов, но нужно сохранять информацию, о предпочтительных магазинах (в которых чаще делают заказы). Магазин имеет несколько отделов, которые характеризуются номером, названием и кол-во журналов (вместимость). Магазин может получать новые журналы и списывать старые. ISBN может измениться в результате переклассификации, а почта в результате перерегистрации. Продавцу могут потребоваться следующие сведения о текущем состоянии библиотеки:

1. Какие журналы были куплены определенным покупателем?
2. Как называется журнал с заданным ISBN?
3. Какой ISBN у журнала с заданным названием?
4. Когда журнал был куплен?
5. Кто из покупателей купил журнал более месяца тому назад?
6. Найти покупателя самых редких журналов (по наличию в магазине)?
7. Какое число покупателей пользуется определенным магазином?
8. Сколько покупателей младше 20 лет?

Выполнение работы.

Для начала необходимо установить зависимости (прописаны в *package.json*) при помощи *nmp i*.

Подключение к базе данных описано в файле *config/database.js*. Для того, чтобы убрать из непосредственно кода программы пароль, была подключена библиотека *dotenv*. Перед запуском программы необходимо создать/отредактировать файл *.env*, указав в нем необходимые параметры:

```
NODE_ENV=development

DB_HOST=host
DB_PORT=port
DB_NAME=name
DB_USER=user
DB_PASS=pass

DB_LOGGING=true
```

Здесь *host* — адрес хоста (т.к. работа происходила в среде WSL Ubuntu, а сама база данных была на этой же машине в среде Windows, использовался шлюз по умолчанию 172.22.0.1), *port* — порт подключения (по умолчанию 5432), *name* — имя базы данных (*lab3*), *user* — имя пользователя (по умолчанию *postgres*), *pass* — пароль.

Сервер описан в файле *server.js*. Реализовано по одному эдпоинту на каждый из запросов (см. раздел Вариант). В запросы 2, 4 и 7 была намеренно введена уязвимость к SQL-инъекциям.

Для начала необходимо заполнить БД данными командой *npm run seed*, после чего запустить сервер командой *npm run start*. После этого можно приступить к тестированию с использованием *sqlmap*. Например, вот так выглядит тестирование первого эндпоинта:

```
sqlmap -u "http://localhost:3000/journals-by-
reader?email=ivan.petrov@email.com" \
--dbms=PostgreSQL \
--risk=3 \
--level=5 \
--batch \
--flush-session
```

Результаты тестирования представлены в таблице 1.\

Таблица 1. Результаты тестирования эндпоинтов.

Тестируемый эндпоинт (соответствует запросу из раздела Вариант)	Результат
1) /journals-by-reader	Уязвимости не найдены
2) /vuln/journal-by-isbn	Обнаружена уязвимость в параметре *isbn*
3) /isbn-by-name	Уязвимости не найдены
4) /vuln/purchases-by-journal-name	Обнаружена уязвимость в параметре *name*
5) /readers-old-purchases	Уязвимости не найдены
6) /buyers-of-rare-journals	Уязвимости не найдены
7) /vuln/readers-by-shop	Обнаружена уязвимость в параметре *shopId*
8) /readers-younger-than-20	Уязвимости не найдены

Все внедренные уязвимости были обнаружены. Причина уязвимостей — использование необработанного пользовательского ввода при формировании SQL-запросов.

Ссылку на исходный код см. в приложении А.

Выводы.

В результате выполнения лабораторной работы были изучены уязвимости БД к SQL-инъекциям. Путь к исправлению — отказ от конкатенации строк в пользу параметризованных запросов (*replacements*/*bind* в Sequelize) или использование ORM-методов (*findOne*, *findAll*, *count* и т.п.), а также общая валидация входных данных и ограничение прав пользователя БД.

ПРИЛОЖЕНИЕ А
ССЫЛКИ НА РЕЗУЛЬТАТЫ РАБОТЫ

Ссылка на Pull-request с результатами выполненной лабораторной работы: https://github.com/moevm/sql-2025-3341/pull/49