Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ

к лабораторной работе №6

на тему

**ЗАЩИТА ОТ АТАКИ МЕТОДОМ ВНЕДРЕНИЯ SQL-КОДА**

Выполнил: студент гр.253505 Сенько Н. С.

Проверил: ассистент кафедры информатики Герчик А.В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc190108682)

[2 Ход работы 4](#_Toc190108683)

[Заключение 5](#_Toc190108684)

# 1 ЦЕЛЬ РАБОТЫ

Цель данной работы заключается в исследовании природы и механизмов возникновения уязвимостей, связанных с SQL-инъекциями, а также в разработке программы, наглядно демонстрирующей уязвимый и защищённый подходы к обработке пользовательского ввода в базу данных.

SQL-инъекции представляют собой один из наиболее распространённых типов атак на веб-приложения. Они позволяют злоумышленникам выполнять произвольные SQL-запросы, извлекать, изменять или удалять данные, а в некоторых случаях — получать полный контроль над базой данных.

В рамках данной работы были созданы две версии программы на языке Python с использованием библиотеки sqlite3 для взаимодействия с базой данных SQLite.

# 2 ХОД РАБОТЫ

В процессе выполнения работы сначала необходимо подготовить программу, чтобы она могла корректно взаимодействовать с базой данных. Для этого запускается *Python*-скрипт в терминале, и программа выводит текущий статус защиты от *SQL*-инъекций, что позволяет сразу понять, в каком режиме она функционирует – защищённом или уязвимом.

Затем, для демонстрации уязвимости, отключается защита, что можно сделать, изменив флаг *PROTECTION\_ENABLED* на *False*. После этого в поля ввода можно ввести специально сформулированную строку, например, «age asc; select \* from users». В уязвимом режиме программа, не проверяя ввод, выполняет запрос, вернув все данные.

После демонстрации уязвимости включается защита путём установки флага *PROTECTION\_ENABLED* в *True*. Повторный ввод той же *SQL*-инъекции теперь не приведёт к выдаче данных – программа выполнит только корректную часть запроса.

В финале анализа сравнивается поведение программы в двух режимах, чтобы на практике оценить эффективность защитных механизмов. На основании наблюдений делается вывод о важности использования подготовленных запросов и правильной обработки входных данных. Такая демонстрация позволяет чётко увидеть, насколько критичны меры предосторожности при работе с пользовательским вводом, и как даже простые методы могут защитить приложение от серьёзных угроз безопасности.

# ЗАКЛЮЧЕНИЕ

В ходе исследования были рассмотрены методы SQL-инъекций, их воздействие на безопасность веб-приложений, а также способы защиты от подобных атак.

Разработка двух версий программы — уязвимой и защищённой — позволила наглядно продемонстрировать, насколько опасны некорректная обработка пользовательских данных и как применение простых, но эффективных методов, таких как параметризованные запросы, может предотвратить потенциальные угрозы.

Результаты исследования указывают на то, что защита от SQL-инъекций является неотъемлемой частью разработки безопасных приложений. Даже базовые меры, такие как проверка вводимых данных и ограничение доступа к базе данных, значительно снижают вероятность успешного взлома.

В целом, исследование подчёркивает важность внимательного подхода к безопасности на всех этапах разработки программного обеспечения. Применение современных методов защиты и регулярное тестирование на уязвимости способствуют созданию надёжных систем, способных противостоять реальным угрозам в условиях постоянно меняющегося ландшафта киберугроз.

def users(ss):

try:

with sqlite3.cnctect('data.db') as cnct:

with cnct:

cur = cnct.cursor()

if IS\_PROTECTED:

if ss and re.match(pattern, ss):

ord = f" ORDER BY {ss}"

else:

ord = " ORDER BY id asc"

q = f'SELECT id, username, age FROM "USERS" {ord}'

cur.execute(q)

users = cur.fetchall()

for user in users:

print(user)

else:

queries = [q.strip() for q in ss.split(';') if q.strip()]

if not queries:

q = 'SELECT id, username, age FROM "USERS" ORDER BY id asc'

cur.execute(q)

users = cur.fetchall()

for user in users:

print(user)

else:

q = f'SELECT id, username, age FROM "USERS" ORDER BY {queries[0]}'

cur.execute(q)

results = cur.fetchall()

print("Результаты:")

for row in results:

print(row)

for q in queries[1:]:

cur.execute(q)

results = cur.fetchall()

if results:

print("Результаты:")

for row in results:

print(row)

else:

print("Запрос выполнен, результатов нет.")

except Exception as e:

print(f"Ошибка при работе с базой данных: {e}")