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

**РЕАЛИЗОВАТЬ ЗАЩИТУ ОТ АТАКИ МЕТОДОМ ВНЕДРЕНИЯ SQL-КОДА**

Выполнил: студент гр.253504

Фроленко К.Ю.

Проверил: ассистент кафедры информатики Герчик А.В.

Минск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc190705396)

[1 ФОРМУЛИРОВКА ЗАДАЧИ 4](#_Toc190705397)

[2 ПРИМЕР ВЫПОЛНЕНИЯ ПРОГРАММЫ 5](#_Toc190705398)

[ЗАКЛЮЧЕНИЕ 6](#_Toc190705399)

ВВЕДЕНИЕ

В современных программных системах безопасность обмена данными и корректная обработка пользовательского ввода являются одними из важнейших аспектов при разработке программного обеспечения. С ростом количества и сложности кибератак вопросы защиты информации приобретают первостепенное значение. Одной из наиболее опасных уязвимостей является SQL-инъекция, когда злоумышленник, используя неэкранированный ввод, может изменить структуру SQL-запроса и получить несанкционированный доступ к базе данных. Такая атака позволяет не только извлечь конфиденциальные данные, но и изменить или удалить их, что может привести к серьёзным последствиям для информационных систем.

Данная работа направлена на демонстрацию принципов защиты от SQL-инъекций. Для реализации эксперимента разработано приложение, которое позволяет сравнить два подхода формирования SQL-запросов. В первом режиме используется безопасный метод – параметризованные запросы, гарантирующие корректную обработку пользовательского ввода, а во втором – небезопасный метод с прямой подстановкой строк, что делает запрос уязвимым к SQL-инъекциям. Такой сравнительный анализ позволяет наглядно увидеть, как простейшие меры контроля ввода могут значительно повысить устойчивость приложения к атакам, а также осознать важность соблюдения принципов безопасного программирования в современных условиях.

1 ФОРМУЛИРОВКА ЗАДАЧИ

Основной задачей данной работы является разработка демонстрационной программы, которая позволяет сравнить два метода формирования SQL-запросов в контексте защиты от SQL-инъекций. Программа должна обеспечивать выбор между режимом, в котором используется параметризованный запрос (безопасный режим), и режимом, где запрос формируется посредством прямой подстановки введённых данных (уязвимый режим).

В ходе эксперимента пользователь вводит строку, представляющую собой имя пользователя. При использовании небезопасного метода происходит непосредственная подстановка этого значения в SQL-запрос. В результате злоумышленник, введя специальный инъекционный код (например, ' OR '1'='1), может изменить логику запроса так, что условие фильтрации всегда оказывается истинным, что приводит к возврату всех записей из таблицы. В защищённом режиме, когда используется подготовленный запрос, введённое значение обрабатывается как литерал, и даже попытка проведения атаки не способна изменить структуру SQL-запроса.

Таким образом, поставленная задача позволяет не только продемонстрировать потенциальные угрозы, связанные с неправильной обработкой пользовательского ввода, но и показать, насколько эффективными могут быть простейшие методы защиты, внедряемые на этапе формирования запросов.

2 ПРИМЕР ВЫПОЛНЕНИЯ ПРОГРАММЫ

При запуске приложения пользователь получает возможность выбора режима работы. На экране выводится приглашение, позволяющее ввести необходимые параметры. В стандартном режиме (с включённой защитой) программа запрашивает имя пользователя и выполняет SQL-запрос с использованием параметризованных выражений, что гарантирует корректное экранирование символов. В случае, если приложение запущено в уязвимом режиме (посредством передачи специального флага, например, --disable-protection), строка, введённая пользователем, подставляется непосредственно в SQL-запрос, что открывает возможность для атаки.

Пример атаки выглядит следующим образом: при отключённой защите пользователь вводит значение ' OR '1'='1.

При подстановке этого значения в запрос получается следующая строка: SELECT \* FROM users WHERE username = '' OR '1'='1'

Условие OR '1'='1' всегда истинно, что приводит к выборке всех записей из таблицы. Таким образом, демонстрация работы приложения в двух режимах наглядно показывает разницу в безопасности.

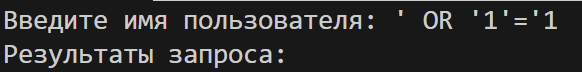


Рисунок 1 – Демонстрация работы приложения в режиме с включённой защитой

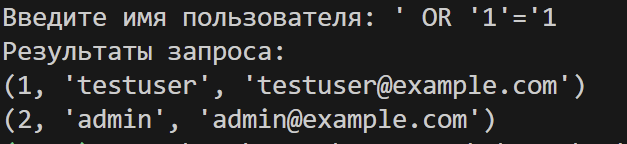


Рисунок 2 – Демонстрация работы приложения в уязвимом режиме (защита отключена)

Приведённый эксперимент показывает, что даже незначительные изменения в способе обработки пользовательского ввода могут иметь критические последствия для безопасности базы данных. Практическая реализация демонстрационного приложения позволяет оценить важность применения современных методов защиты в реальных условиях эксплуатации.

# ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы было разработано и протестировано демонстрационное приложение, позволяющее исследовать проблему SQL-инъекций и оценить эффективность применения параметризованных запросов для защиты от подобных атак. Экспериментальное тестирование показало, что использование небезопасного метода формирования SQL-запросов с прямой подстановкой пользовательского ввода приводит к полной компрометации запроса, что может позволить злоумышленнику получить несанкционированный доступ ко всем данным. Напротив, применение подготовленных выражений эффективно защищает систему от SQL-инъекций, поскольку введённые данные обрабатываются как литералы, а не как часть SQL-кода.

Проведённое исследование подтверждает, что даже простейшие меры по защите пользовательского ввода оказывают существенное влияние на безопасность приложения. Данный сравнительный анализ демонстрирует, насколько критично правильно организовать обработку данных в условиях постоянно растущей угрозы кибератак. Наглядное сопоставление результатов работы приложения в двух режимах подчёркивает необходимость использования проверенных методов защиты, что является залогом создания надёжных и устойчивых к атакам информационных систем.

Таким образом, выполненная работа не только демонстрирует теоретические аспекты проблемы SQL-инъекций, но и подтверждает практическую значимость реализации механизмов защиты на этапе разработки программного обеспечения. В условиях современной информационной среды соблюдение принципов безопасного программирования является необходимым условием для успешного функционирования любых систем.

ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг программного кода**

import psycopg2

import sys

def main():

enable\_protection = True

if "--disable-protection" in sys.argv:

enable\_protection = False

username = input("Введите имя пользователя: ")

try:

conn = psycopg2.connect(

dbname="lab6", user="postgres", password="8025", host="localhost"

)

cur = conn.cursor()

if enable\_protection:

query = "SELECT \* FROM users WHERE username = %s"

cur.execute(query, (username,))

else:

query = f"SELECT \* FROM users WHERE username = '{username}'"

cur.execute(query)

rows = cur.fetchall()

print("Результаты запроса:")

for row in rows:

print(row)

except Exception as e:

print("Ошибка:", e)

finally:

if "cur" in locals():

cur.close()

if "conn" in locals():

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

# ' OR '1'='1