

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Факультет безопасности информационных технологий

Дисциплина:

«Информационная безопасность баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«SQL-инъекции»

Выполнили:

Чу Ван Доан, студент группы N3247



(подпись)

Проверил:

Волков А.Г.

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

Содержание	3
1 SQL-инъекции	5
1.1 Цель работы.....	5
1.2 Задание.....	5
1.3 Ход работы.....	5
1.3.1 Задание 1	5
1.3.2 Задание 2	6
1.3.3 Задание 3	6
Заключение	8

1 SQL-инъекции

1.1 Цель работы

Получение навыков SQL-инъекций.

1.2 Задание

1.2.1 Задание 1: В случае объединения таблиц покажите, как злоумышленник может узнать количество столбцов второй таблицы.

1.2.2 Задание 2: Предложите подход для получения структуры базы данных (включая название столбцов таблицы).

1.2.3 Задание 3: Покажите пример использования подготовленных параметров для вашего любимого языка программирования.

1.3 Ход работы

1.3.1 Задание 1


Создадим таблицу students.

```
n33471_16_lab6=# CREATE TABLE students (id BIGSERIAL PRIMARY KEY, name VARCHAR(30), age INT, country VARCHAR(50));
CREATE TABLE
n33471_16_lab6=# INSERT INTO students (name, age, country) VALUES ('huan', 23, 'VietNam'),
n33471_16_lab6=# ('Yulianna', 21, 'Russia'), ('Zhao Lu Si', 22, 'China');
INSERT 0 3
n33471_16_lab6=# SELECT * FROM students;
 id |   name   | age | country
----+-----+----+-----
  1 | huan     |  23 | VietNam
  2 | Yulianna |  21 | Russia
  3 | Zhao Lu Si | 22 | China
(3 rows)
```

Рисунок 1 – Создание таблицы students

```
n3247_22_lab6=# create table count_column (value int);
CREATE TABLE
```

Рисунок 2 – Создание таблицы count_column

аем количество столбцов таблицы с помощью information_schema.columns.

```
n3247_22_lab6=# select value from count_column union
select count(column_name) from information_schema.columns where table_name = 'students';
 value
-----
      4
(1 row)
```

Рисунок 3 – Поиск количества column

1.3.2 Задание 2

С помощью `information.columns`, мы можем узнать структуру базы данных.

```
n33471_16_lab6=# SELECT value1, value2 FROM data_struct UNION
n33471_16_lab6=# SELECT column_name, data_type FROM information_schema.columns WHERE table_name = 'students';
 value1 |      value2
-----+-----
 country | character varying
 id       | bigint
 age      | integer
 name     | character varying
(4 rows)
```

Рисунок 4 – Подход для получения структуры базы данных

1.3.3 Задание 3

Создадим таблицу `info_user`.

```
n33471_16_lab6=# CREATE TABLE info_user (id BIGSERIAL, username VARCHAR(20), password VARCHAR(20));
CREATE TABLE
n33471_16_lab6=# INSERT INTO info_user (username, password) VALUES ('user1', 'passworduser1'), ('user2', 'passworduser2');
n33471_16_lab6=# ('user2', 'passworduser2');
INSERT 0 2
n33471_16_lab6=# SELECT * FROM info_user;
 id | username | password
---+---+---
  1 | user1    | passworduser1
  2 | user2    | passworduser2
(2 rows)
```

Рисунок 5 – Создание таблицы `info_user`

```

GNU nano 7.2 lab5_3.py
python
import psycopg2
import getpass
# Подключение к базе данных
connection = psycopg2.connect(
    database="n33471_16_lab6",
    user="postgres",
    password="mynewpassword",
    host="localhost",
    port="5432"
)
# Создание курсора
cursor = connection.cursor()
# Данные для вставки
username = str(input("Input username: "))
password = str(input("Input password: "))
# Подготовленный запрос с использованием параметров
query = "SELECT * FROM info_user WHERE username=%s AND password=%s;"
# Выполнение подготовленного запроса с данными
cursor.execute(query, (username, password))
# Распечатать результаты
result = cursor.fetchone()
print(result)
# Закрытие курсора и соединения
cursor.close()
connection.close()

```

Рисунок 6 – Код

```

(root@kali)-[/home/kali]
# python3 lab5_3.py
Input username: user1 or 1=1
Input password: passworduser1
None

(root@kali)-[/home/kali]
# python3 lab5_3.py
Input username: user1
Input password: passworduser1
(1, 'user1', 'passworduser1')

(root@kali)-[/home/kali]
# python3 lab5_3.py
Input username: user2
Input password: passworduser2
(2, 'user2', 'passworduser2')

```

Рисунок 7 – Результат использования подготовленных параметров для Python

```
(root@kali)-[/home/kali]
# python3 lab5_3.py
Input username: user1
Input password: passworduser1 or 1=1
None

n33471_16_lab6=# \q
(root@kali)-[/home/kali]
# python3 lab5_3.py
Input username: user2
Input password: passworduser2 or 1+1=2
None
```

Рисунок 8 – Результат использования подготовленных параметров для Python

В этом примере мы используем библиотеку 'psycopg2' для подключения к PostgreSQL, создаём курсор и выполняем подготовленный запрос с использованием параметров ('%s'). Затем мы передаем кортеж с данными для вставки ('username' и 'password') в качестве второго аргумента метода 'execute'. Это предотвращает SQL-инъекцию и делает код более читаемым и безопасным.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы был изучен теоретический материал по SQL-инъекциям. Приобретенные знания были применены на практике в СУБД PostgreSQL.