МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №34

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, к.т.н. |  |  |  | В.А. Мыльников |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6  **Защита базы данных от SQL - инъекций** |
|  |
| по курсу: БЕЗОПАСНОСТЬ СИСТЕМ БАЗ ДАННЫХ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 3843 |  |  |  | А.П.Конева |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**Цель работы**

Получение навыков по защите баз данных от SQL-инъекций.

**Задача**

Разработать и продемонстрировать модель защиты объектов базы данных от SQL-инъекций.

SQL-инъекция (внедрение SQL-кода) – один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.

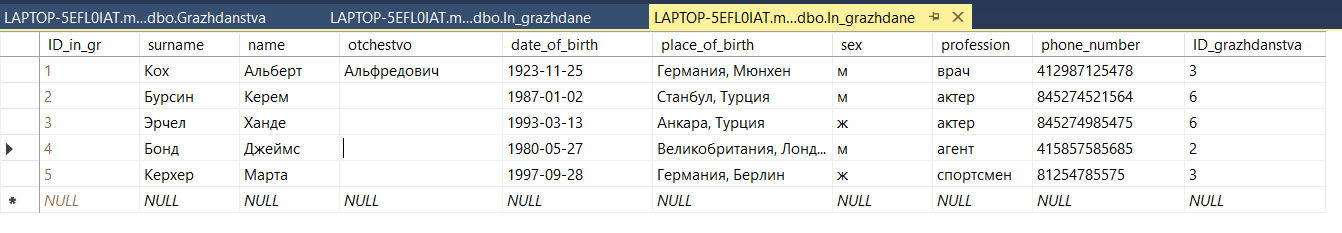
Для выполнения данной лабораторной работы был использован MS SQL Server. Для демонстрации рассматриваемых SQL-инъекции выбрали таблицу In\_grazhdane в БД migreg(Рис.1)

Рисунок 1 - Таблица In\_grazhdane в БД migreg

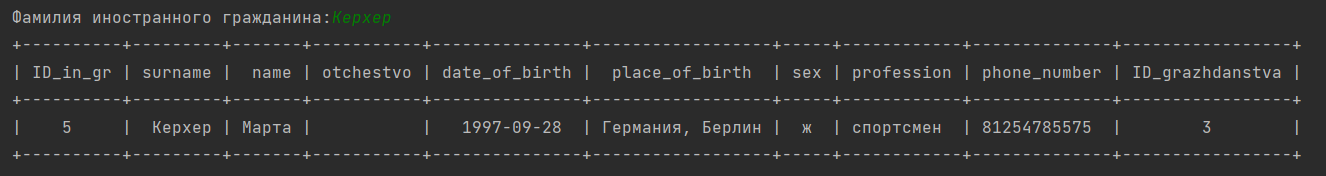
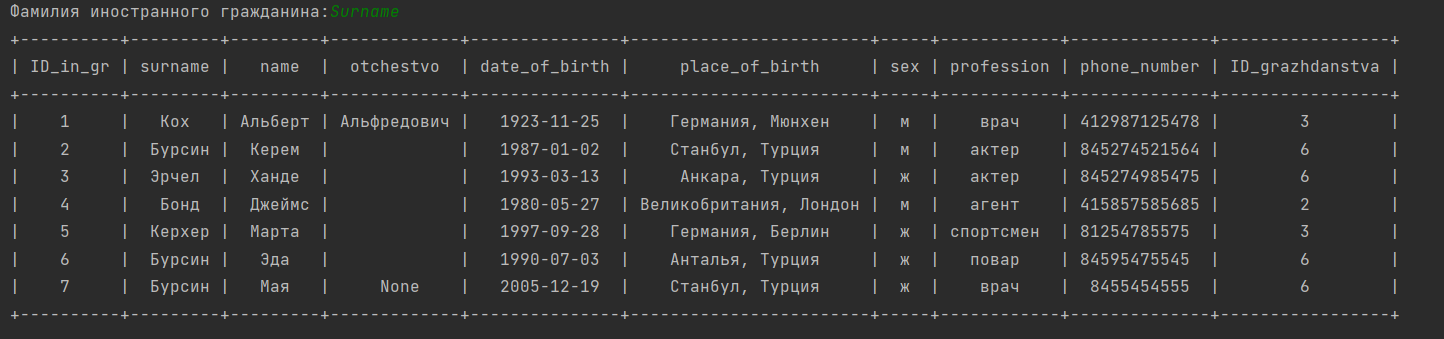
Для демонстрации реализации SQL-инъекций и методов защиты от них разработали консольное приложение, с помощью которого можно взаимодействовать с БД. Для работы приложения необходимо ввести нужную «Фамилию иностранного гражданина». Демонстрация работы консольного приложения в результате выполнения SQL - запроса «SELECT \* FROM In\_grazhdane WHERE Surname = Керхер» приведена на Рис.2

Рисунок 2 - Демонстрация работы консольного приложения

**Реализация SQL-инъекций**

Рассмотрим реализацию *SQL-инъекции, основанной на операциях сравнения, всегда возвращающих ИСТИНУ*. Поскольку field=field всегда ИСТИННО, то результатом такого запроса будет выборка всех записей выбранной таблицы. Например, результат реализации SQL-инъекции «SELECT \* FROM In\_grazhdane WHERE Surname = Surname» изображен на Рис.3.

Рисунок 3 - Результат SQL-инъекции, основанной на операции сравнения, всегда возвращающей ИСТИНУ

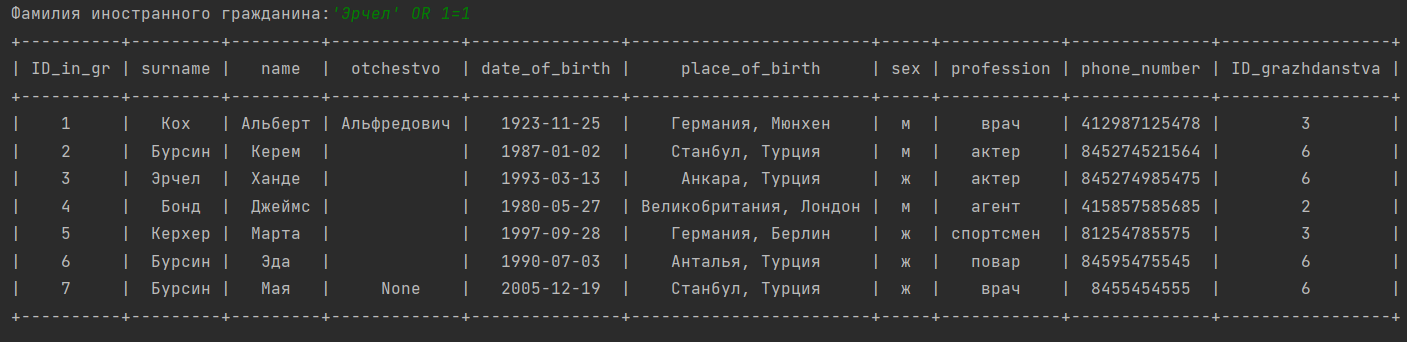
Рассмотрим реализацию *SQL - инъекции, основанной на применении логического оператора OR в запросе*. Результат реализации SQL-инъекции «SELECT \* FROM In\_grazhdane WHERE Surname = ‘Эрчел’ OR 1=1» изображен на Рис.4 и также представляет собой выборку всех записей выбранной таблицы.

Рисунок 4 - Результат SQL - инъекции, основанной на применении логического оператора OR

Рассмотрим результат выполнения следующего запроса на Рис.5:

"SELECT \* FROM In\_grazhdane WHERE Surname = '" + surname + "' AND sex = 'ж'"

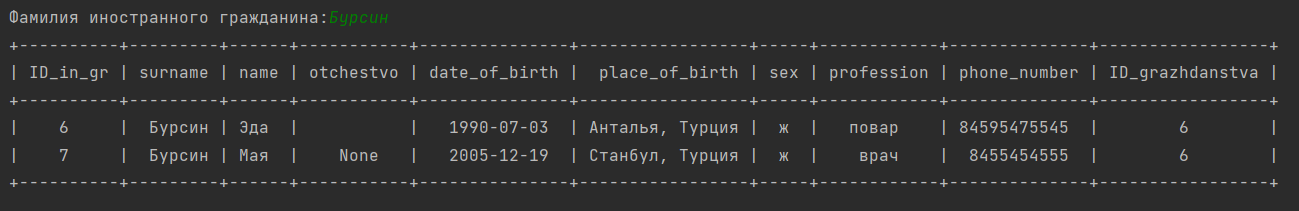
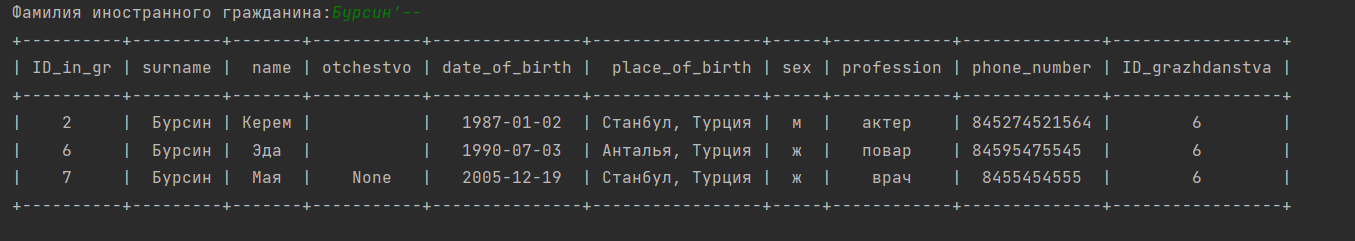


Рисунок 5 - Результат выполнения запроса

При введении фамилии иностранного гражданина из таблицы выбираются записи, соответствующие гражданам с фамилией Бурсин женского пола.

Проверим результат реализации *SQL - инъекции, основанной на применении комментариев внутри запроса*. После выполнения SQL-инъекции «SELECT \* FROM In\_grazhdane WHERE Surname = ‘Бурсин’-- AND sex=’ж’» получаем следующее (Рис.6):

Рисунок 6 - Результат SQL - инъекции, основанной на применении комментариев внутри запроса

Такой результат получился по причине того, что была введена фамилия вместе с закрывающей кавычкой и комментарием, что позволяет буквально отключает условия выборки, указанные далее. Иными словами, полученная в результате реализации инъекции выборка представляет из себя список всех граждан с фамилией Бурсин. Иначе говоря, итоговый запрос к таблице получился следующим:

«SELECT \* FROM In\_grazhdane WHERE Surname ='Бурсин'»

**Защита от SQL - инъекций**

Для защиты от подобных SQL-инъекций нужно использовать параметризированный запрос:

'''SELECT \* FROM In\_grazhdane WHERE Surname = ? ''', surname

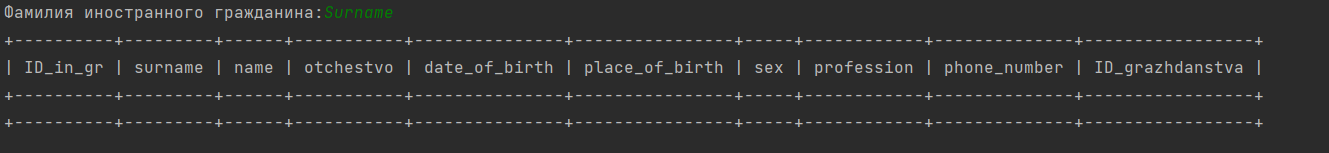
Тогда в случае *SQL-инъекции, основанной на операциях сравнения, всегда возвращающих ИСТИНУ*: поскольку значение «Surname» не совпало ни с одной из фамилий иностранных граждан, данный параметризированный SQL-запрос вернет пустой результат (Рис.7):

Рисунок 7 - Демонстрация защиты от SQL-инъекции, основанной на операции сравнения, всегда возвращающей ИСТИНУ

Далее, в случае *SQL - инъекции, основанной на применении логического оператора OR в запросе:* поскольку значение «'Эрчел' OR 1=1» не совпало ни с одной из фамилий иностранных граждан, тот же параметризированный SQL-запрос также вернет пустой результат (Рис.8):

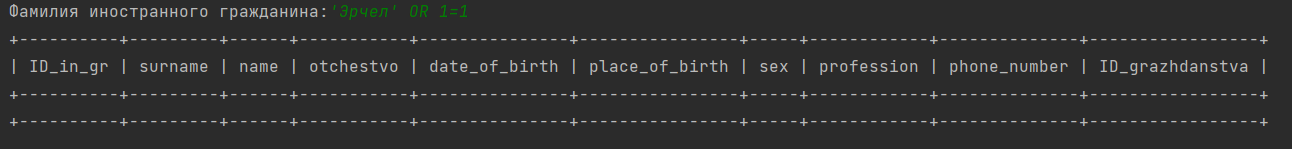


Рисунок 8 - Демонстрация защиты от SQL-инъекции, основанной на применении логического оператора OR в запросе

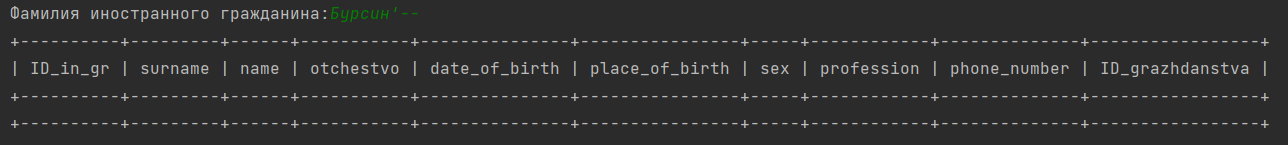
Далее, в случае *SQL - инъекции, основанной на применении комментариев внутри запроса*: поскольку значение «Бурсин'--» не совпало ни с одной из фамилий иностранных граждан, тот же параметризированный SQL-запрос также вернет пустой результат (Рис.9):

Рисунок 9 - Демонстрация защиты от SQL-инъекции, основанной на применении комментариев внутри запроса

**Вывод**

Исследовали методы защиты баз данных от SQL-инъекций. Разработали консольное приложение для взаимодействия с БД. Результаты реализации SQL-инъекций отображены на Рис. 3,4,6. Продемонстрировали защиту от SQL -инъекций с использованием метода параметризации SQL- запросов (Рис. 7 - 9).

**Листинг программы**

**ББД6.py**

from prettytable import PrettyTable, from\_db\_cursor  
import pyodbc  
  
mytable = PrettyTable()  
conn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"  
 "Server=LAPTOP-5EFL0IAT;"  
 "Database=migreg;"  
 "Trusted\_Connection=yes;")  
c = conn.cursor()  
surname = input("Фамилия иностранного гражданина:")  
c.execute('''SELECT \* FROM In\_grazhdane WHERE Surname = ? ''', surname)

mytable = from\_db\_cursor(c)  
print(mytable)  
conn.close()