|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**Лабораторная работа №6**

**«Оценка безопасности web-страниц и приложений с использованием ручного и автоматизированного анализа наличия уязвимостей типа “SQL Injection”»**

**ДИСЦИПЛИНА: «Защита информации»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-72Б | |  |  | ( | Сафронов Н.С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Ерохин И.И. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

Калуга, 2023

**Цель работы:** освоение и систематизация знаний об уязвимостях и инструментах их выявления.

**Постановка задачи**

1. Написать собственный web-сайт или приложение, которое будет уязвимо для SQL-инъекций.
2. Проверить его уязвимость.
3. Исправить уязвимость и убедиться, что уязвимости больше нет.

**Листинг уязвимого приложения**

import sqlite3

from flask import Flask, redirect, render\_template, session, url\_for, request

app = Flask(\_\_name\_\_)

app.secret\_key = "A0Zr98j/3yX R~XHH!jmN]LWX/,?RT"

@app.route("/")

def index():

if "username" in session:

return render\_template("index.html")

return redirect(url\_for("auth"))

@app.route("/auth")

def auth():

if "username" in session:

return redirect(url\_for("index"))

return render\_template("auth.html")

@app.route("/api/auth", methods=["POST"])

def authenticate():

user = request.form["user"]

password = request.form["pass"]

conn = sqlite3.connect("db/data.sqlite")

cursor = conn.cursor()

cursor.execute(

"SELECT COUNT(\*) FROM users WHERE name = '%s' AND pass = '%s'"

% (

user,

password,

)

)

res = cursor.fetchone()

conn.close()

if res[0] != 0:

session["username"] = request.form["user"]

return redirect(url\_for("index"))

return redirect(url\_for("auth"))

@app.route("/logout")

def logout():

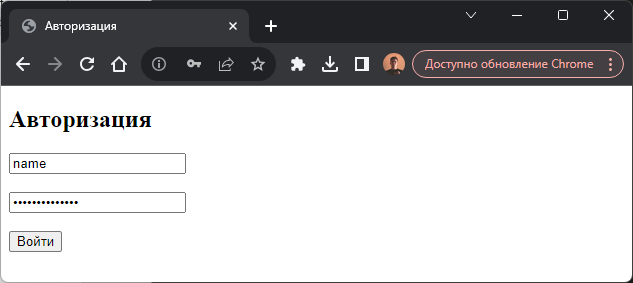
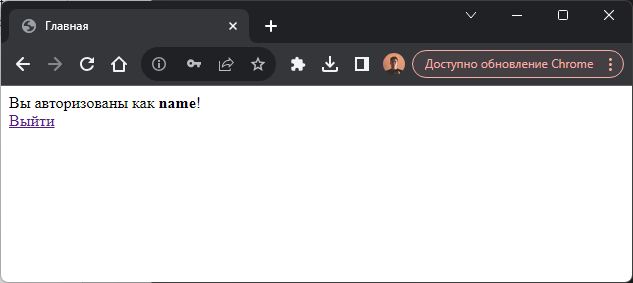
session.pop("username", None)

return redirect(url\_for("auth"))

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

app.run("127.0.0.1", 80, True)

**Результаты выполнения работы**

**Рисунок 1 –** Тестирование SQL-инъекции строкой вида «- hi' or 1=1—»

Протестируем другие инъекции:

* hi' or 1=1-- - проходит
* ' or 1=1- - проходит
* " or 1=1-- - проходит
* or 1=1- - проходит
* ' or 'a'='a - проходит
* " or "a"="a - проходит
* ') or ('a'='a - проходит

**Листинг изменённого приложения**

import sqlite3

from flask import Flask, redirect, render\_template, session, url\_for, request

app = Flask(\_\_name\_\_)

app.secret\_key = "A0Zr98j/3yX R~XHH!jmN]LWX/,?RT"

@app.route("/")

def index():

if "username" in session:

return render\_template("index.html")

return redirect(url\_for("auth"))

@app.route("/auth")

def auth():

if "username" in session:

return redirect(url\_for("index"))

return render\_template("auth.html")

@app.route("/api/auth", methods=["POST"])

def authenticate():

user = request.form["user"]

password = request.form["pass"]

conn = sqlite3.connect("db/data.sqlite")

cursor = conn.cursor()

cursor.execute(

"SELECT COUNT(\*) FROM users WHERE name = ? AND pass = ?",

(user,

password)

)

res = cursor.fetchone()

conn.close()

if res[0] != 0:

session["username"] = request.form["user"]

return redirect(url\_for("index"))

return redirect(url\_for("auth"))

@app.route("/logout")

def logout():

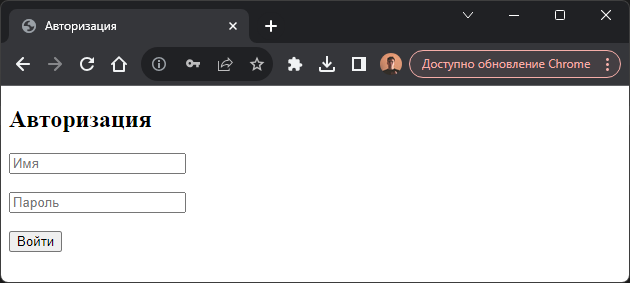
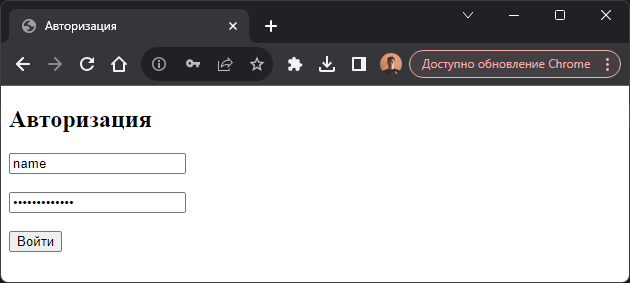
session.pop("username", None)

return redirect(url\_for("auth"))

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

app.run("127.0.0.1", 80, True)

**Результаты выполнения работы**



**Рисунок 2 –** Тестирование SQL-инъекции строкой вида «- hi' or 1=1—»

Протестируем другие инъекции:

* hi' or 1=1-- - не проходит
* ' or 1=1- - не проходит
* " or 1=1-- - не проходит
* or 1=1- - не проходит
* ' or 'a'='a - не проходит
* " or "a"="a - не проходит
* ') or ('a'='a - не проходит

**Вывод**: в ходе выполнения лабораторной работы были освоены и систематизированы знания об уязвимостях и инструментах их выявления.