Министерство Образования, Культуры и Исследований

Молдавский Государственный Университет

Факультет Математики и Информатики

Департамент Информатики

**Лабораторная работа №1**

По курсу: Безопасность web приложений.

На тему: Методы предотвращения SQL-injection в веб приложениях.

Выполнил: Гуцу Даниил,

группа IA2102

Проверила: Плешка Наталья

Кишинев, 2024

# **Требования к лабораторной работе:**

1. Создайте 7 новых записей в таблице «user».

2. Попробуйте использовать SQL-инъекцию, чтобы войти в панель администрирования (закрытое пространство администратора), не использовав правильные данные существующих аккаунтов (посмотрите дополнительно https://www.owasp.org/index.php/Testing\_for\_SQL\_Injection\_(OTG-INPVAL-005))

3. Создайте client-side и server-side скрипты, которые будут бороться с возможностью ввода кода SQL, в элементы управления формы, путем проверки данных.

4. Проверьте безопасность созданного мини-приложения.

**Что такое SQL-инъекция?**

SQL-инъекция — это техника атаки, при которой злоумышленник использует недостатки безопасности входных данных веб-приложения, чтобы внедрить SQL-код непосредственно в SQL-запросы, выполняемые приложением. Это может привести к раскрытию, изменению или удалению данных в базе данных, а также к получению несанкционированного доступа к системе.

**Как бороться с SQL-инъекциями?**

* **Параметризация запросов (Prepared Statements):**

Используйте параметризованные запросы, в которых значения параметров отделяются от SQL-запроса. Это позволяет избежать внедрения SQL-кода через пользовательский ввод.

* **Использование хранимых процедур (Stored Procedures):**

Применяйте хранимые процедуры, предварительно созданные в базе данных. Они уменьшают риски SQL-инъекций, так как не позволяют внедрять SQL-код напрямую.

* **Ограничение прав доступа (Least Privilege):**

Предоставляйте минимально необходимые права доступа к базе данных пользователям и приложениям. Это помогает ограничить возможные последствия SQL-инъекций.

* **Экранирование символов (Escape Characters):**

Экранируйте специальные символы, такие как одинарные кавычки (') и двойные кавычки (") во входных данных, чтобы предотвратить их неправильную интерпретацию как SQL-код.

* **Валидация и фильтрация ввода (Input Validation and Sanitization):**

Проводите валидацию входных данных на стороне сервера и клиента, разрешая только допустимые символы и форматы. Фильтруйте пользовательский ввод от потенциально опасных символов.

* **Обновление и обновление системы (Regular Updates and Patching):**

Регулярно обновляйте и патчите СУБД и приложения для исправления уязвимостей, включая те, которые могут быть использованы для SQL-инъекций.

* **Обучение и осведомленность (Education and Awareness):**

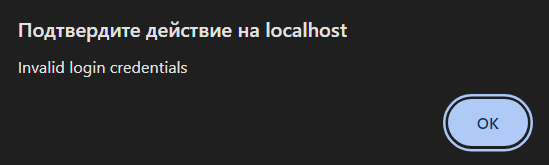
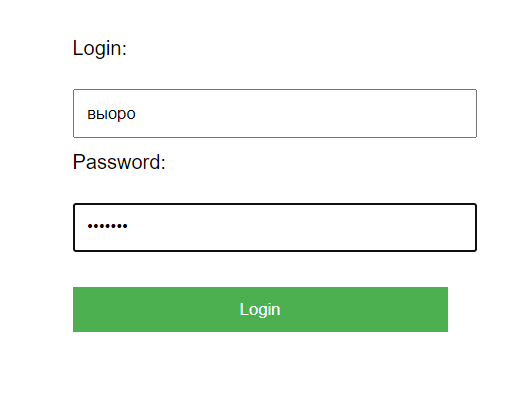
Обучайте разработчиков, тестировщиков и администраторов баз данных о методах предотвращения и обнаружения SQL-инъекций, а также общих методах безопасности.

Применяя сочетание этих методов, можно существенно снизить риски SQL-инъекций и обеспечить безопасность базы данных и приложения.

# **Попытки ввода кодов SQL**

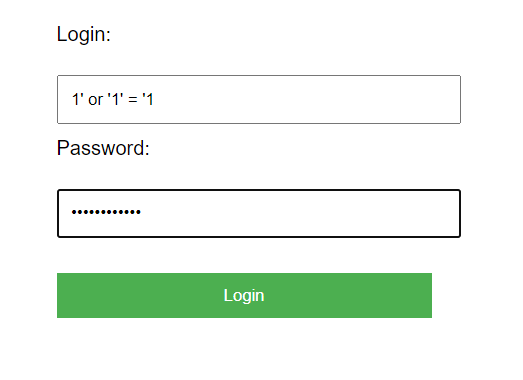
Обеспечив безопасность от SQL инъекций своему приложению, проведем тест системы на ожидаемую безопасность.

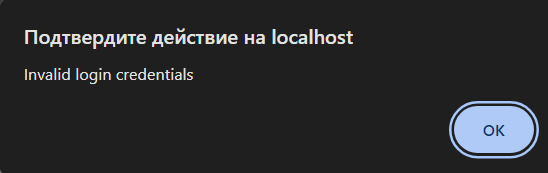
1. Неправильный ввод данных



Этот тест успешно пройден. При подтверждении ввода несуществующих данных пользователь возвращается на ту же страницу, где должен ввести данные повторно.

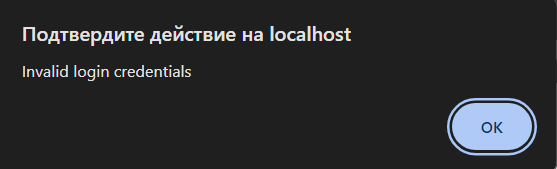
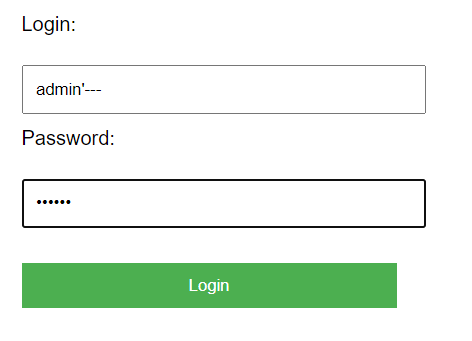
1. Ввод логин = 1' or '1' = '1; пароль = 1' or '1' = '1, которые конкатенируются к запросу и запрос все время будет возвращать «true»





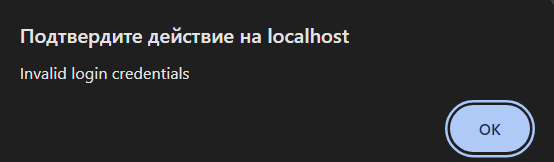
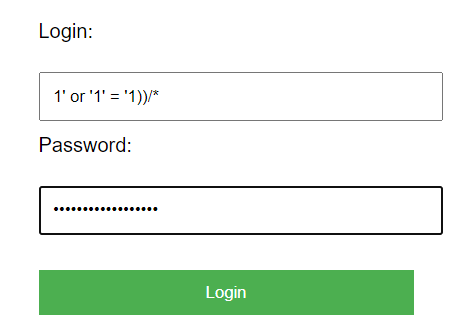
Тест успешно пройден.

1. Комментируется дальнейший текст запроса (-- ввод комментария в SQL)



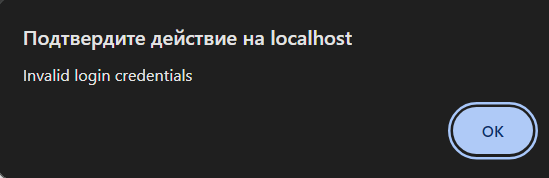
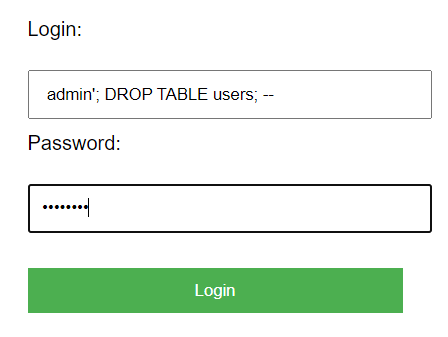
Тест успешно пройден.

1. Ввод комментария в PHP



Тест успешно пройден.

1. Ввод не соответствующий паттернам на клиентской части.



Последний тест пройден успешно.

**Как обеспечена безопасность в приложении?**

**Client-side**

function validateForm() {

            var login = document.getElementById("login").value;

            var password = document.getElementById("password").value;

            // Проверка на наличие запрещенных символов для SQL-запросов

            var sqlPattern = /[**\'\"\\**]/g;

            if (sqlPattern.test(login) || sqlPattern.test(password)) {

                alert("Login and password must not contain special characters");

                return false;

            }

            if (login == "" || password == "") {

                alert("Login and password must be filled out");

                return false;

            }

        }

**Server-side**

$login = sanitize($\_POST['login']);

$password = sanitize($\_POST['password']);

$sql = "SELECT \* FROM user WHERE login='$login' AND password='$password'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

    $\_SESSION['loggedin'] = true;

    header("Location: admin\_panel.php");

    exit;

} else {

    echo "<script>alert('Invalid login credentials');</script>";

    echo "<script>window.location.href='login.php';</script>";

    exit;

}

<?php

function sanitize($input) {

    // Защита от SQL инъекций и XSS атак

    $input = trim($input);

    $input = stripslashes($input);

    $input = htmlspecialchars($input);

    return $input;

}

?>

# **Вывод**

В ходе лабораторной работы я научился обеспечивать безопасность моему приложению от SQL-инъекций, используя различные методы, рекомендуемые owasp.