# Лабораторная работа №2 Исследование уязвимостей ПО и методов их устранения

1. Цель работы
   1. Изучить основные угрозы безопасности веб-приложений, такие как SQL-инъекции, XSS-инъекции, а также методы защиты от них;
   2. Научиться применять защитные меры в разработке безопасных веб-приложений.
2. Литература
   1. Зверева В. П., Сопровождение и обслуживание программного обеспечения компьютерных систем : учебник для студ. учреждений сред. проф. Образования / В. П. Зверева, А. В. Назаров. – М. : Издательский центр «Академия», 2018. – 256 с.
3. Подготовка к работе
   1. Повторить теоретический материал (см. п.2).
   2. Изучить описание лабораторной работы.
4. Основное оборудование
   1. Персональный компьютер.
5. Задание
   1. Используя приложение п. 9, реализовать и запустить приложение, подверженное уязвимостям;
   2. Протестировать SQL-инъекции п.9.2, выполнить авторизацию без учетных данных, исправить код приложения для устранения данных уязвимостей;
   3. Протестировать XSS-инъекции п.9.3, выполнить инъекцию кода перенаправляющего пользователя на другую страницу после загрузки, исправить код приложения для устранения данных уязвимостей;
   4. Протестировать уязвимости авторизации п.9.4, изменить почту другого пользователя, исправить код приложения для устранения данных уязвимостей.
6. Порядок выполнения работы
   1. Повторить теоретический материал п. 3.1;
   2. Исследовать уязвимости веб-приложений ПО п. 5.1-5.4;
   3. Ответить на контрольные вопросы п. 8;
   4. Заполнить отчет п. 7.
7. Содержание отчета
   1. Титульный лист;
   2. Цель работы;
   3. Код исправленных модулей п. 5.3;
   4. Ответы на контрольные вопросы п. 6.3;
   5. Вывод по проделанной работе.
8. Контрольные вопросы
   1. Какие методы используются для защиты от sql-инъекций?
   2. Какие методы используются для защиты от xss-инъекций?
9. Приложение
   1. Настройка базы данных

Для начала создадим базу данных и таблицы, необходимые для работы приложения.

CREATE DATABASE lab\_security;

USE lab\_security;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

CREATE TABLE comments (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

comment TEXT NOT NULL

);

-- Добавляем пользователя для тестирования

INSERT INTO users (username, password, email) VALUES ('admin', 'password', 'example@mail.ru');

INSERT INTO users (username, password, email) VALUES ('user', 'password', 'other@mail.ru');

CREATE USER 'lab\_security'@'localhost' identified with mysql\_native\_password BY 'lab\_security';

GRANT ALL PRIVILEGES ON lab\_security.\* TO 'lab\_security'@'localhost';

* 1. Реализация уязвимого приложения

Форма входа (login.php)

Этот код демонстрирует уязвимость к SQL-инъекциям при вводе учетных данных.

<?php

$conn = new mysqli('localhost', 'lab\_security', 'lab\_security', 'lab\_security');

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$username = $\_POST['username'];

$password = $\_POST['password'];

$query = "SELECT \* FROM users WHERE username = '$username' AND password = '$password'";

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

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

echo "Добро пожаловать, $username!";

} else {

echo "Неверное имя пользователя или пароль.";

}

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<form method="POST" action="login.php">

Username: <input type="text" name="username"><br>

Password: <input type="password" name="password"><br>

<input type="submit" value="Login">

</form>

</body>

</html>

* 1. Форма комментариев (comments.php)

Этот код демонстрирует уязвимость к XSS, позволяя пользователям вставлять произвольный HTML или JavaScript код в поле комментария.

<?php

$conn = new mysqli('localhost', 'lab\_security', 'lab\_security', 'lab\_security');

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$username = $\_POST['username'];

$comment = $\_POST['comment'];

$query = "INSERT INTO comments (username, comment) VALUES ('$username', '$comment')";

$conn->query($query);

}

$result = $conn->query("SELECT \* FROM comments");

?>

<!DOCTYPE html>

<html>

<head>

<title>Comments</title>

</head>

<body>

<h1>Comments</h1>

<form method="POST" action="comments.php">

Username: <input type="text" name="username"><br>

Comment: <textarea name="comment"></textarea><br>

<input type="submit" value="Post Comment">

</form>

<h2>All Comments</h2>

<ul>

<?php while ($row = $result->fetch\_assoc()): ?>

<li><strong><?php echo $row['username']; ?>:</strong> <?php echo $row['comment']; ?></li>

<?php endwhile; ?>

</ul>

</body>

</html>

* 1. Форма смены пароля (update-email.php)

Этот код демонстрирует уязвимость авторизации, позволяя пользователям изменять чужие учетные данные.

<?php

$conn = new mysqli('localhost', 'lab\_security', 'lab\_security', 'lab\_security');

// Форма обновления email

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$username = $\_POST['username'];

$new\_email = $\_POST['new\_email'];

$query = "UPDATE users SET email = '$new\_email' WHERE username = '$username';" ;

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

if ($result > 0) {

echo "Email for user $username updated to $new\_email";

}

}

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Update Email</title>

</head>

<body>

<form action="update-email.php" method="post">

<label for="username">User Name:</label>

<input type="text" id="username" name="username" required>

<label for="new\_email">New Email:</label>

<input type="email" id="new\_email" name="new\_email" required>

<input type="submit" value="Update Email">

</form>

</body>

</html>