

# Web Application Penetration Testing Report Of Juice Shop

Done by **Lia Potikyan**



## Scope

Scope	Scope Type	Start Date	End Date
http://localhost:3000	Web Application Penetration Testing	March 30, 2024	April 2, 2024

## Description

This report presents the findings of a comprehensive web application penetration testing conducted on Juice Shop. The objective of this assessment was to identify and analyze security vulnerabilities within the application, assess their potential impact, and provide recommendations for remediation.

Discovered Vulnerabilities

Vulnerability Name	Severity	Status
1. SQL Injection	CRITICAL	VULNERABLE
2. Sensitive Data Exposure	HIGH	VULNERABLE
3. Weak Hashing Algorithm	HIGH	VULNERABLE
4. DOM XSS	MEDIUM	VULNERABLE
5. XSS	MEDIUM	VULNERABLE
6. Insecure Direct Object Reference	LOW	VULNERABLE

# DOM XSS

## VULNERABLE

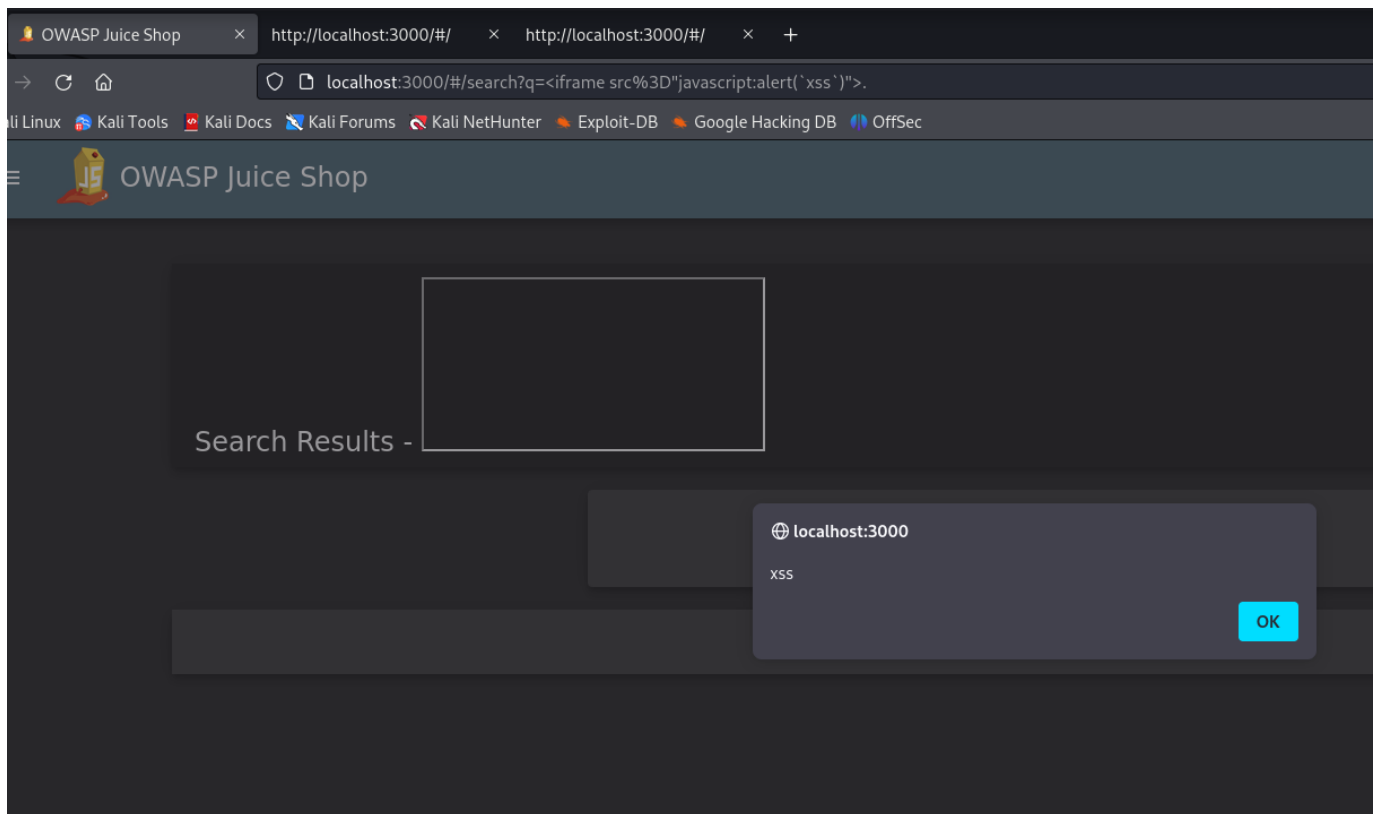
A DOM-based Cross-Site Scripting (XSS) attack occurs when an attacker is able to inject malicious scripts into a web application, which are then executed in the victim's browser. In DOM-based XSS attacks, the vulnerability arises from the client-side code (usually JavaScript) manipulating the Document Object Model (DOM) in an unsafe manner.

### Used payload:

```
<iframe src="javascript:alert(`xss`)">
```

### Proof Of Concept:

As can be seen, the user is able to inject and execute the javascript code into the page.



# XSS (Cross-Site Scripting)

**VULNERABLE**

The XSS vulnerability arises from the application's failure to properly sanitize user-supplied input, allowing attackers to inject malicious JavaScript code into the application's output. In this specific case, the payload was successfully injected into the application and executed in the victim's browser when the vulnerable page is accessed.

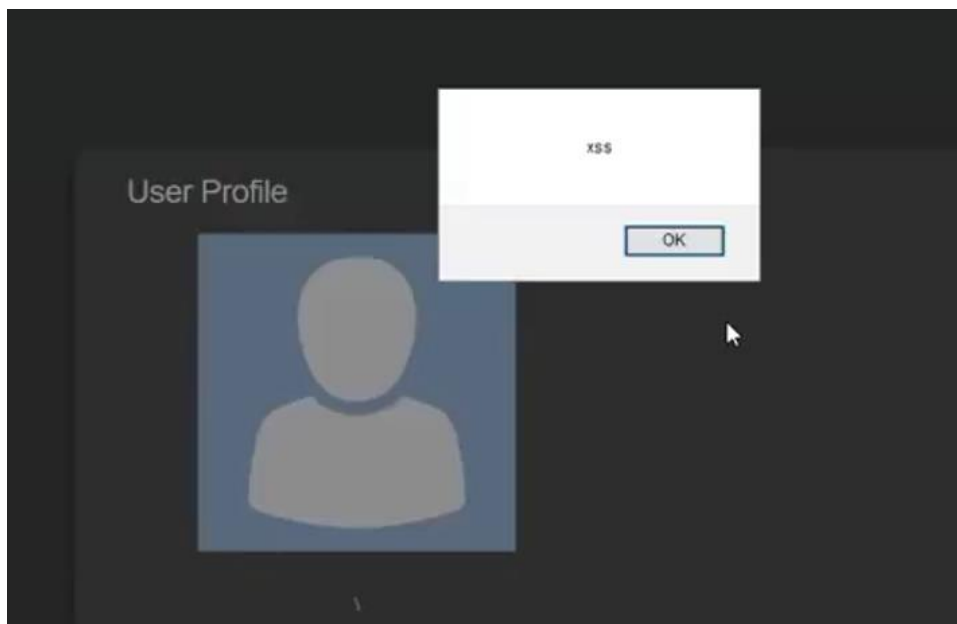
Impact:

- **Data Theft:** Attackers can steal sensitive information (e.g., session cookies, user credentials) from authenticated users by executing malicious scripts in their browsers.
- **Session Hijacking:** Attackers can hijack user sessions by stealing session tokens and impersonating legitimate users.
- **Phishing Attacks:** Attackers can create convincing phishing pages within the application to trick users into disclosing sensitive information or performing malicious actions.

Used payload:

```
<script>alert('xss')</script>
```

Proof Of Concept:



**Solution:**

- Implement input validation and output encoding to sanitize user-supplied input and prevent injection of malicious scripts.
- Implement Content Security Policy (CSP) headers to restrict the sources from which scripts can be executed, mitigating the impact of XSS attacks.

# SQL Injection

**VULNERABLE**

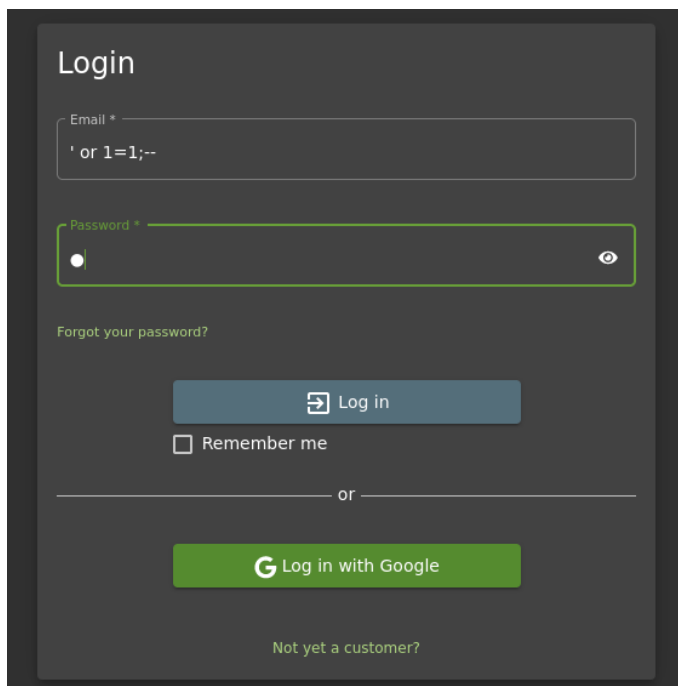
SQL injection is a type of security vulnerability commonly found in web applications. It occurs when an attacker is able to manipulate input data sent to an application's SQL database, allowing them to execute arbitrary SQL commands.

**Used payload:**

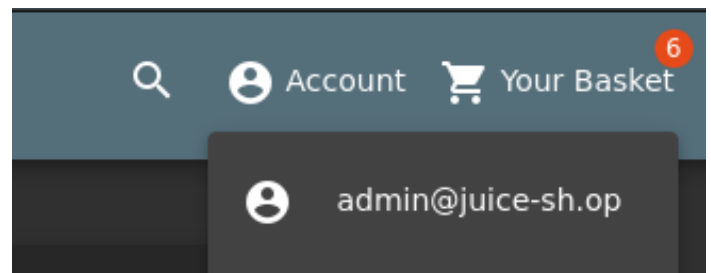
```
' or 1=1;--
```

**Proof Of Concept:**

Logging as Admin



The screenshot shows a login form titled "Login". It has two input fields: "Email \*" and "Password \*". The "Email \*" field contains the payload "' or 1=1;--". Below the email field is a link "Forgot your password?". There are two buttons: "Log in" and "Log in with Google". Below the "Log in" button is a checkbox labeled "Remember me". At the bottom, there is a link "Not yet a customer?".



**Solution:**

- Implement parameterized queries or prepared statements to sanitize user input and prevent SQL injection attacks.
- Utilize input validation to ensure that user-supplied data conforms to expected formats and values.

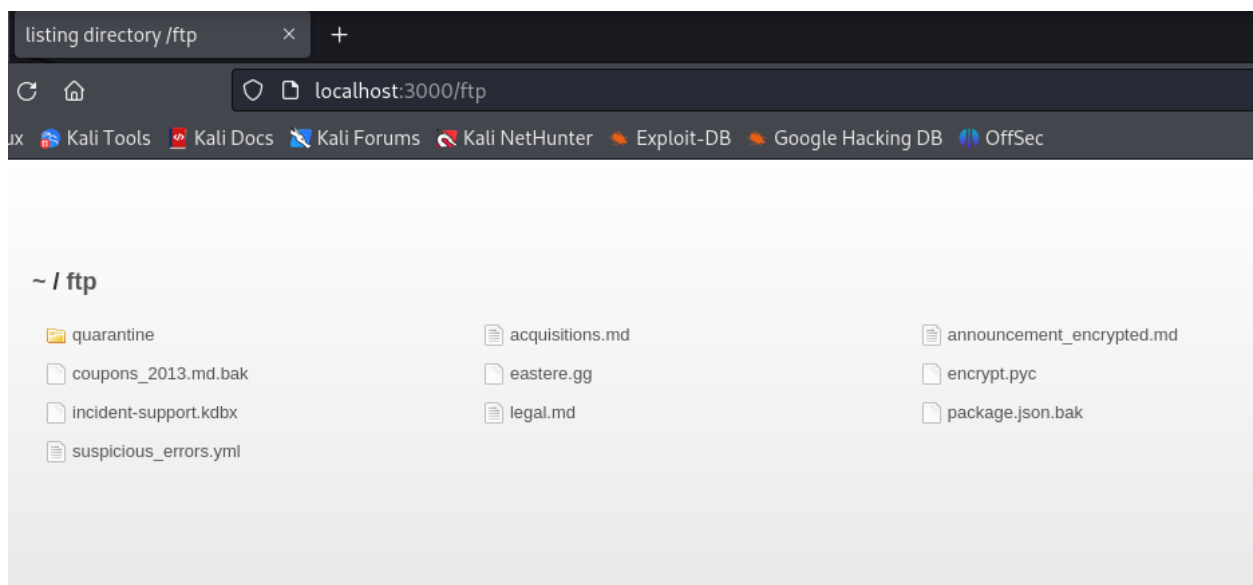
# Sensitive Data Exposure

**VULNERABLE**

Information disclosure, also known as information leakage, occurs when a website unintentionally exposes sensitive information to its users. In various contexts, websites might inadvertently disclose a range of sensitive data to potential attackers, including:

- Personal data of other users, such as usernames or financial information
- Confidential commercial or business data
- Technical specifics regarding the website and its underlying infrastructure

## Proof Of Concept:



## Solution:

- Encrypt sensitive data at rest and in transit to protect against unauthorized access and disclosure.
- Implement access controls and authentication mechanisms to restrict access to sensitive information based on user privileges.



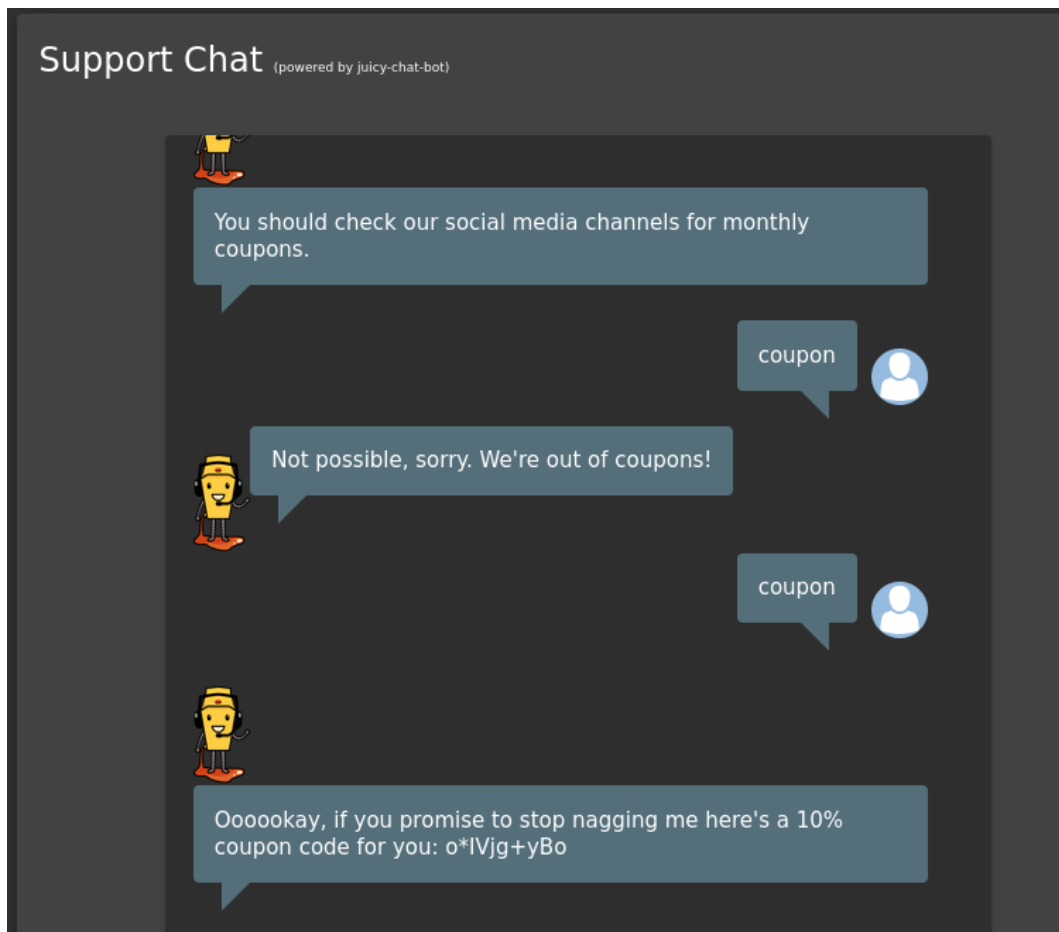
# Insecure Direct Object

**VULNERABLE**

## Reference

The chatbot's functionality allows users to access or manipulate resources (such as messages or user data) without proper authorization checks, an attacker could exploit this vulnerability to access sensitive information or perform unauthorized actions. In this case, the vulnerability might manifest when attempting to overload the chatbot with multiple messages, potentially revealing sensitive information or causing unexpected behavior due to inadequate input validation or authorization controls.

### Proof Of Concept:



**Solution:**

- Implement proper access controls and authorization checks to ensure that users can only access and manipulate resources for which they are authorized.
- Validate user input and implement adequate input validation and sanitization to prevent manipulation of object references.

# Weak Hashing Algorithm

**VULNERABLE**

The Juice Shop application was found to utilize the MD5 (Message Digest Algorithm 5) hashing algorithm for cryptographic operations, particularly in the context of password hashing and storage. MD5 is a cryptographic hash function that is known to be vulnerable to various attacks, including collision attacks, preimage attacks, and length extension attacks. As a result, the use of MD5 for cryptographic purposes poses significant security risks to the application.

- **Data Integrity Compromise:** The susceptibility of MD5 to collision attacks undermines the integrity of cryptographic operations, allowing attackers to manipulate data without detection.
- **Authentication Weakness:** Vulnerabilities in MD5 can be exploited to bypass authentication mechanisms or forge digital signatures, compromising the security of user accounts.
- **Password Security Risk:** If MD5 is used for password hashing, it exposes user passwords to potential compromise in the event of a breach, as attackers can leverage preimage attacks to reverse-hash passwords.

## **Solution:**

- Replace the usage of MD5 hashing algorithm with modern and secure hash functions (e.g., SHA-256, bcrypt) for password hashing and storage.
- Employ salting and key stretching techniques to enhance password security and resilience against brute-force attacks.

**Lia Potikyan**

**Liapotikyan006@gmail.com**