## Bug Report - Juice Shop Exploratory Testing - Security

**Date:** Dec 29, 2024

**Tester:** Aathiraja

**Application:** Juice Shop (<https://juice-shop.herokuapp.com/#/> )

**Severity:** High

**Vulnerability:** Potential SQL Injection Vulnerability

**Description:**

* During exploratory testing, the search functionality within Juice Shop was observed.
* By entering malicious input, such as single quotes ('), double quotes ("), semicolons (;), or other special characters into the search field, the application might be susceptible to SQL injection attacks.
* If the application does not properly sanitize and validate user input before constructing SQL queries, an attacker could potentially manipulate the query to:
  + Retrieve sensitive data from the database.
  + Modify or delete data within the database.
  + Gain unauthorized access to the application's backend.

**Steps to Reproduce:**

1. Access the Juice Shop application.
2. Navigate to the product listing page.
3. Enter malicious input (e.g., ' OR 1=1 -- ) into the search field.
4. Observe the application's behavior.

**Example**:

curl --location '127.0.0.1:5000/client\_login' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--data-urlencode 'userName=test123" OR "1"="1' \

--data-urlencode 'email=test123@test.com" OR "1"="1' \

--data-urlencode 'password=test@1234" OR "1"="1'

**Expected Behavior:**

* The application should gracefully handle invalid input and display an appropriate error message.
* The application should not execute any unintended SQL queries.

**Actual Behavior:**

* The application may exhibit unexpected behavior, such as:
  + Displaying unintended results.
  + Returning database errors.
  + Crashing or becoming unresponsive.

**Impact:**

* Potential data breaches, including customer data, financial information, and internal application data.
* System instability and disruption of service.
* Unauthorized access to sensitive information.

**Recommendation:**

* Implement proper input validation and sanitization for all user-supplied input, particularly in search queries.
* Use parameterized queries or prepared statements to prevent SQL injection attacks.
* Regularly review and update application security best practices.

**Note:**

* This report is based on exploratory testing and further investigation and exploitation attempts to confirm the presence and severity of the vulnerability given in the security test report.