

# TASK - 1

## Web Application Security Assessment Report

**Application:** OWASP Juice Shop

**Testing Tool:** OWASP ZAP

**Date of Assessment:** 29 July 2025

**Assessor:** Mohith S

**CIN:** FIT/JUL25/CS2578

---

### 1. Overview

This document presents the results of a security evaluation carried out on the OWASP Juice Shop web application using OWASP ZAP. The assessment aimed to detect exploitable weaknesses, measure the risks they pose, and propose countermeasures to improve security resilience.

---

### 2. Purpose of Assessment

The primary objectives of this security review were:

- To replicate real-world attack vectors in a controlled manner.
  - To identify security flaws present in the application.
  - To align findings with the OWASP Top 10 threat categories.
  - To deliver remediation guidance that can reduce exposure to attacks.
- 

### 3. Scope

- **Application Under Test:** OWASP Juice Shop (hosted locally)
  - **Primary Tool:** OWASP ZAP (Zed Attack Proxy)
  - **Vulnerability Areas:** Injection attacks, security misconfigurations, information exposure, and related web risks.
-

## 4. Testing Methodology

A phased approach was followed to ensure comprehensive coverage:

1. **Environment Setup & Reconnaissance**
    - Deployed Juice Shop on localhost.
    - Configured browser traffic through OWASP ZAP proxy.
  2. **Passive Reconnaissance**
    - Reviewed HTTP requests and responses.
    - Checked for missing headers, information leakage, and cache weaknesses.
  3. **Active Testing**
    - Performed automated scans for threats like XSS, CSRF, and header injection.
    - Verified notable results through manual exploration.
  4. **Reporting**
    - Recorded tool outputs, evidence, and screenshots.
    - Mapped each vulnerability with severity ratings and recommendations.
- 

## 5. Tools Utilized

- **OWASP Juice Shop:** Intentionally insecure app designed for training and research.
  - **OWASP ZAP:** Open-source proxy and vulnerability scanner.
  - *(Optional)* **Kali Linux:** Supporting environment containing security testing utilities.
- 

## 6. Risk Classification

Each issue was prioritized using the following categories:

- **High:** Critical weakness, highly exploitable.
  - **Medium:** Moderate-level risk requiring some effort to exploit.
  - **Low:** Minor issue, limited potential impact.
  - **Informational:** No direct threat but could aid in further attacks.
- 

## 7. Findings Summary

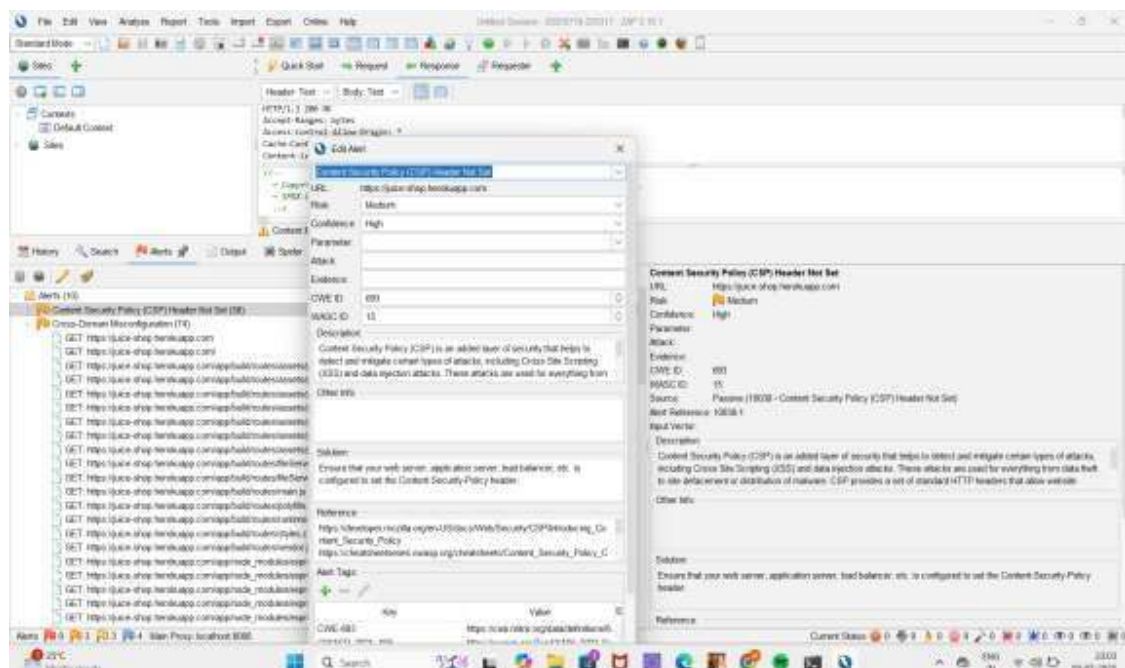
The test revealed multiple **medium** and **low** severity issues within the Juice Shop application. While the platform is intentionally insecure, the results closely resemble weaknesses commonly

found in production systems. Strengthening HTTP headers, enforcing strict access policies, and improving content controls are necessary steps to enhance overall application security.

## 8. Sample Finding

### Finding: Missing Content Security Policy (CSP) Header

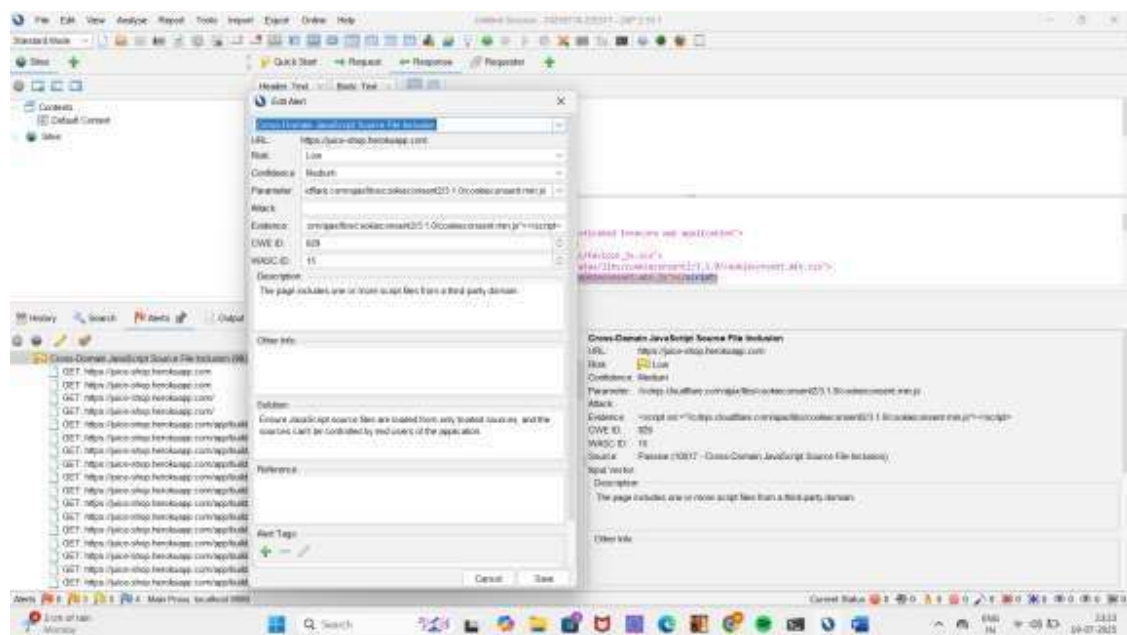
- **Severity:** Medium
- **Description:** The application does not return a CSP header, leaving browsers unable to restrict untrusted content sources.
- **Impact:** Attackers may inject malicious scripts, leading to theft of session data or execution of unauthorized actions.
- **OWASP Category:** A5 – Security Misconfiguration



**Explanation:**

**Impact:**

## OWASP Mapping: A1 – Injection



## Strict-Transport-Security Header Not Set

**Risk Level:** Medium

**Description:** Missing HSTS header allows connections over insecure HTTP. This exposes users to man-in-the-middle (MITM) attacks.

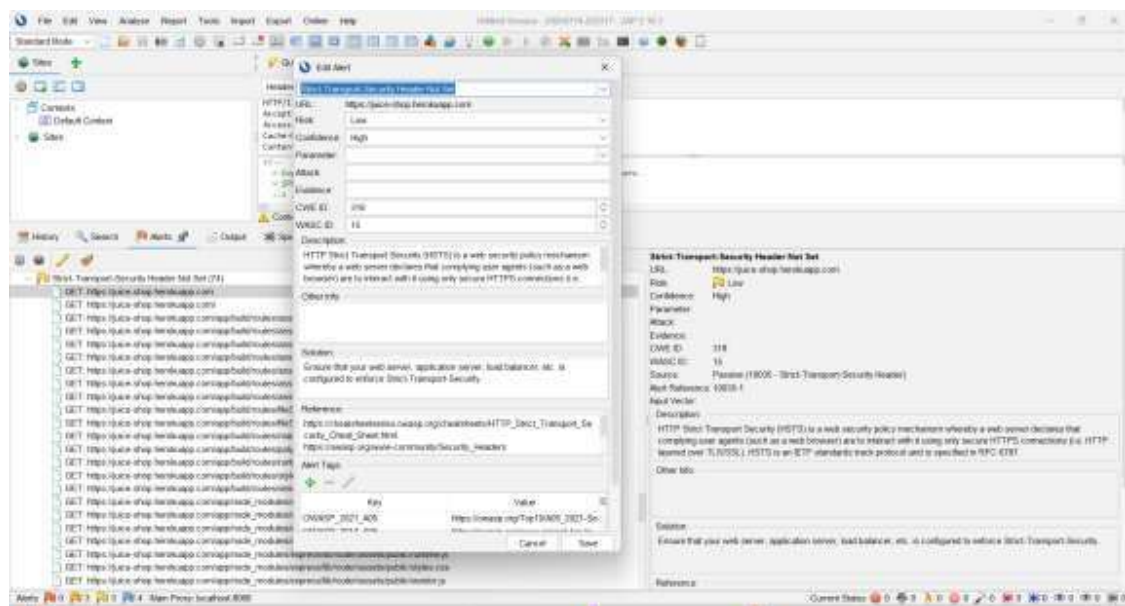
**Explanation:**

The application doesn't enforce HTTPS using the Strict-Transport-Security (HSTS) header.

**Impact:**

Leaves users open to **Man-in-the-Middle (MITM)** attacks over unsecured networks.

**OWASP Mapping:** A5 – Security Misconfiguration



# Timestamp Disclosure - Unix

**Risk Level:** Low

**Description:** Timestamps disclosed in HTTP responses may help attackers to determine server software versions or other information useful in targeted attacks.

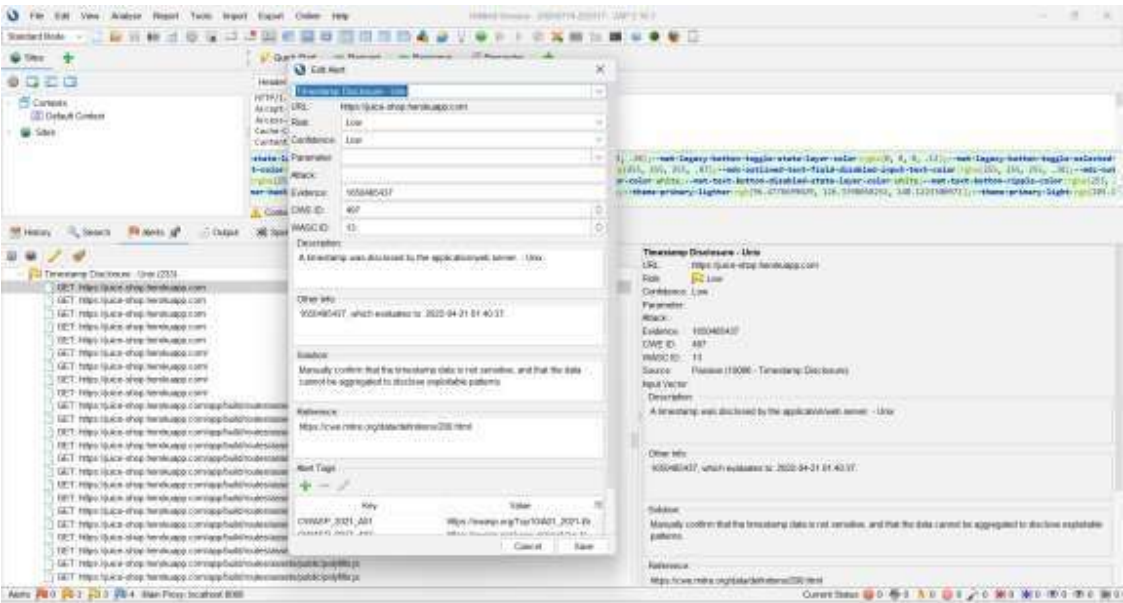
**Explanation:**

Unix timestamps are present in responses or URLs, potentially revealing information about backend systems.

**Impact:**

Helps attackers profile the application, servers, or session generation patterns.

**OWASP Mapping:** A6 – Sensitive Data Exposure



# Information Disclosure - Suspicious Comments

**Risk Level:** Low

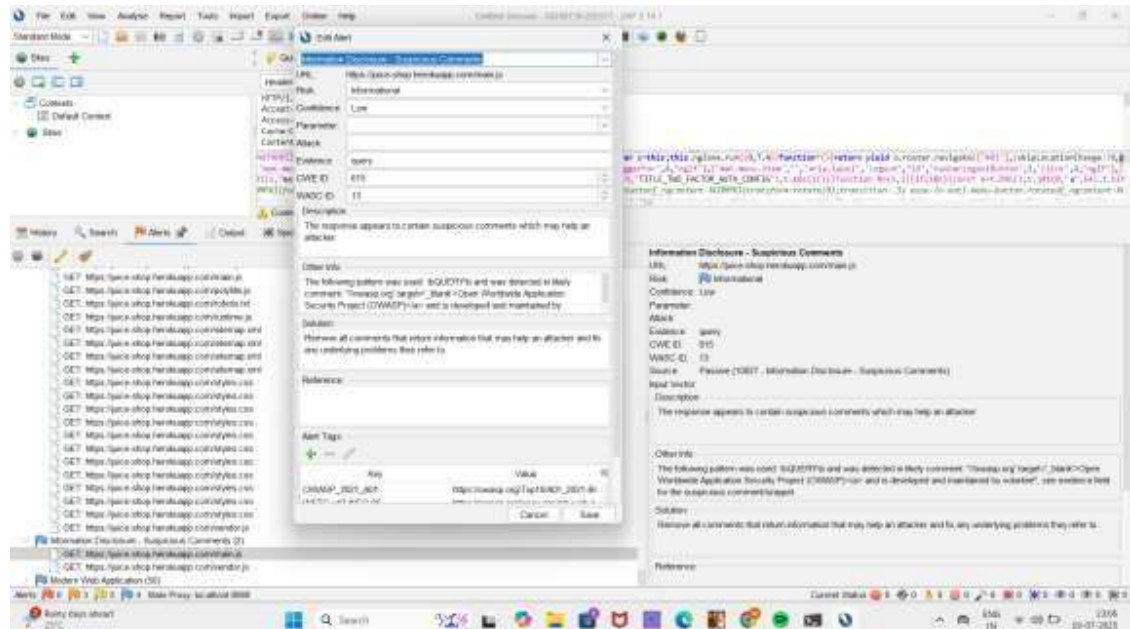
**Description:** Source code contains suspicious or debug-related comments. These might reveal sensitive internal information to an attacker.

**Explanation:**

Developer comments are visible in the source code.

**Impact:**

These may reveal debugging information, credentials, or internal logic useful to attackers

**OWASP Mapping: A6 – Sensitive D Exposure**

ata



## Modern Web Application

**Risk Level:** Informational

**Description:** General detection of a modern web application. This is not necessarily a vulnerability but useful information for testers.

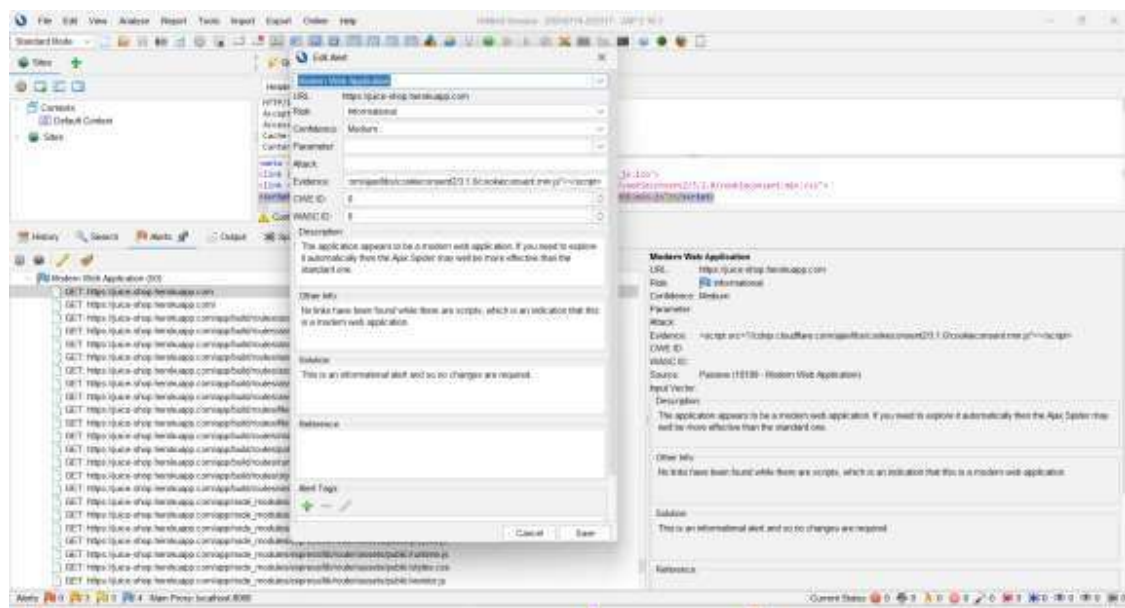
**Explanation:**

Indicates the application is built using a modern frontend framework.

**Impact:**

No direct security impact. Helps understand the technology stack used.

**OWASP Mapping:** Not Applicable



## Hidden File Found

**Risk Level:** Medium

**Description:** Sensitive hidden file was found accessible (\_darcs). This may leak credentials or configuration data.

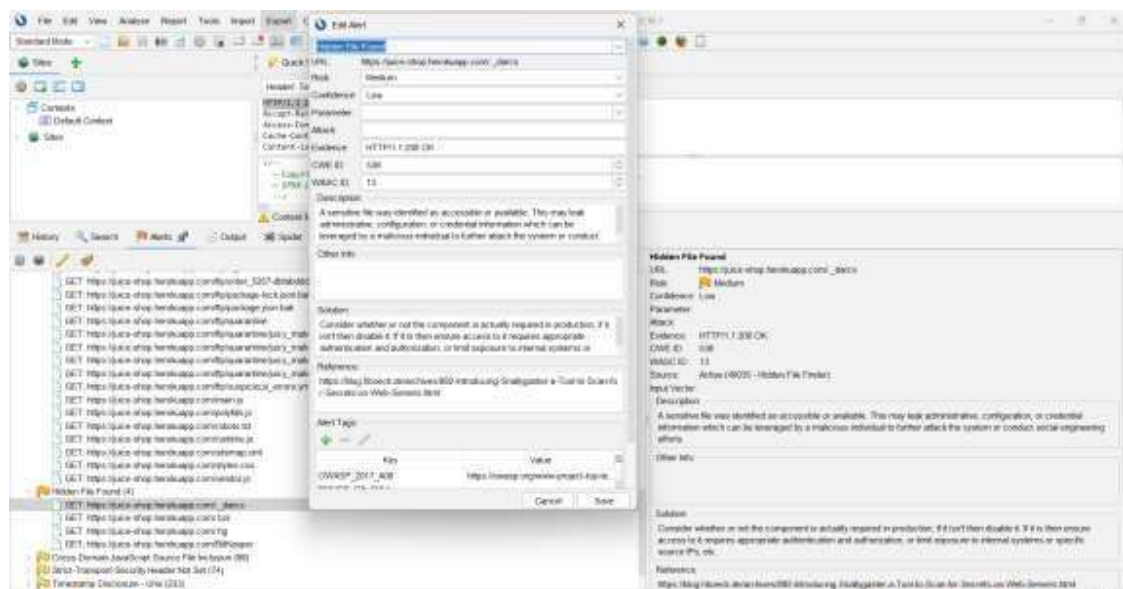
**Explanation:**

A hidden file (\_darcs) was discovered, which might contain configuration or history.

**Impact:**

Could leak credentials, source code, or internal configuration.

**OWASP Mapping:** A5 – Security Misconfiguration



## User Agent Fuzzer

**Risk Level:** Medium

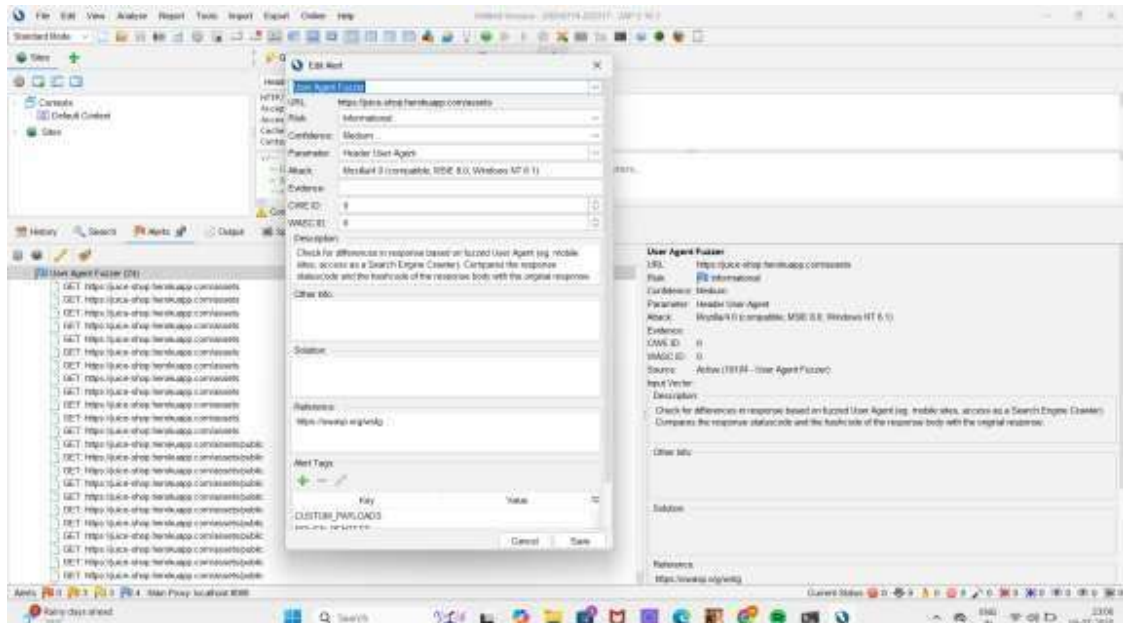
**Description:** Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and the hashcode of the response body with the original response.

**Explanation:**

The application responded differently when accessed with modified User-Agent headers.

**Impact:**

Can be used to fingerprint the application or bypass features.

**OWASP Mapping: A9 – Insufficient Logging & Monitoring**

## Re-examine Cache-control Directives

**Risk Level:** Low

**Description:** The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resources should be reviewed to ensure that no sensitive content will be cached.

### Explanation:

Improper or missing Cache-Control headers allow sensitive content to be cached.

### Impact:

Sensitive data might be stored in shared/public caches (e.g., proxy servers, browsers).

**OWASP Mapping:** A5 – Security Misconfiguration

The screenshot displays the Burp Suite application window. On the left, the 'History' tab is active, showing a list of HTTP requests. The selected request is a GET request to 'https://space-shop.herokuapp.com/appmode\_modules/...'. The main panel shows the details of this request, including the 'Cache-control' header. The 'Quick Scan' tab is also visible, showing the 'Re-examine Cache-control Directives' vulnerability. The 'Description' field states: 'The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resource should be reviewed to ensure that no sensitive content will be cached.' The 'Solution' field provides advice: 'For sensitive content, ensure the cache-control HTTP header is set with "no-cache, no-store, must-revalidate". If an asset should be cached consider setting the directives "public, max-age, immutable".' The 'References' field lists links to OWASP and other resources. The 'Risk' is listed as 'Informational' and the 'Confidence' as 'Low'.

## Vulnerability: SQL Injection – Authentication Bypass

### Location:

Login Page – `/#/login`

---

### Description:

An SQL Injection vulnerability was discovered on the login page of the OWASP Juice Shop application. The application fails to properly validate and sanitize user inputs before incorporating them into SQL queries. As a result, an attacker can inject malicious SQL code into the input fields to alter the logic of the SQL query.

This vulnerability allowed the login authentication to be bypassed, granting unauthorized access to the application without valid credentials.

---

### Attack Vector:

#### Payload used:

```
vbnet
Copy code
' OR 1=1--
```

- Injected into the **email/username field**
- Any value (e.g., `abc`) was entered as the password
- Upon submitting the form, the application logged the user in successfully

---

### Impact:

- Bypass of login authentication
- Unauthorized access to user or administrator accounts
- Potential access to sensitive data
- Increased risk of privilege escalation and data breaches

---

### OWASP Top 10 Mapping:

- **A03:2021 – Injection**

---

### Recommended Remediation:

- Use **parameterized queries (prepared statements)** instead of dynamic SQL.
- Validate and sanitize all user inputs on both client-side and server-side.






- Avoid directly incorporating user input into database queries.
- Implement a **Web Application Firewall (WAF)** to monitor and block injection attempts.
- Log and monitor suspicious login behavior for early detection.

---

Evidence:

- Screenshot of the login form with the injection payload
- Screenshot showing successful login after payload submission

The screenshot displays a web application interface. At the top, a green notification banner reads: "You successfully solved a challenge: Web3 Sandbox (Find an accidentally deployed code sandbox for writing smart contracts on the fly.)" with a close button (X). Below this, a dark gray login form is centered. The form has a title "Login". It contains two input fields: "Email\*" and "Password\*". The "Email\*" field contains the text "' OR 1=1--". The "Password\*" field contains a series of dots and has toggle icons for visibility. Below the password field is a link "Forgot your password?". There is a "Log in" button with a key icon, a "Remember me" checkbox, and a horizontal line with the word "or" in the center. Below this is a "Log in with Google" button with the Google logo. At the bottom of the form is a link "Not yet a customer?".


 OWASP Juice Shop   



You successfully solved a challenge: Web3 Sandbox (Find an accidentally deployed code sandbox for writing smart contracts on the fly.) X




You successfully solved a challenge: Login Admin (Log in with the administrator's user account.) X

You successfully solved a challenge: Confidential Document (Access a confidential document.) X

All Products



 OWASP Juice Shop


' OR '1'='1'   

You successfully solved a challenge: Web3 Sandbox (Find an accidentally deployed code sandbox for writing smart contracts on the fly.) X

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.) X

You successfully solved a challenge: Confidential Document (Access a confidential document.) X

Search Results - ' OR '1'='1



No results found

Try adjusting your search to find what you're looking for.