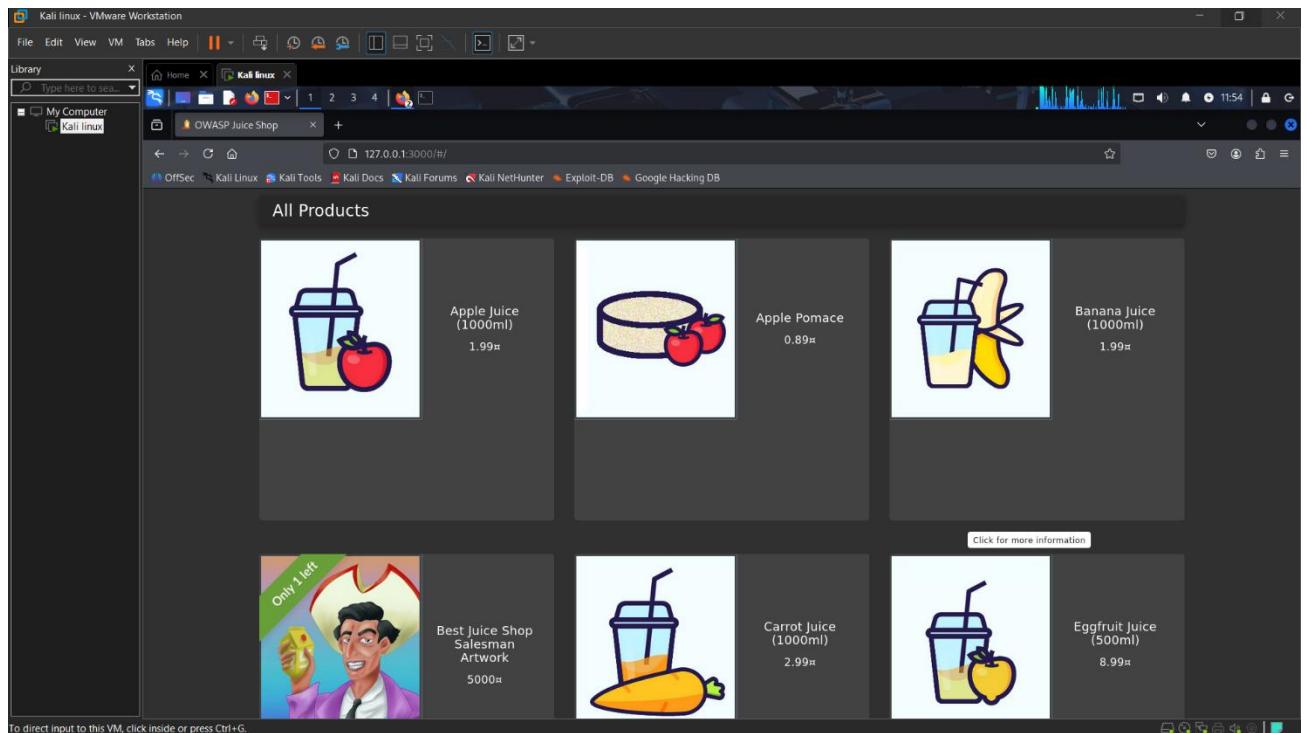


Web Application Vulnerability Testing

Target: OWASP Juice Shop

Tool: Burp Suite Community Edition

Environment: Kali Linux (Localhost)



Vulnerability 1:

A1-Broken Access Control.

The application allows users to access resources which all are belong for other users. Proper authorization checking is missing on server side.

Affected Endpoint- Get /rest/basket/{id}

Steps:

1. Login as a normal user.
2. Intercept the request /rest/basket/6 using Burp Suite
3. Change the basket ID from 6 to 5
4. Forward the request

Result:

The server returned another user's basket details with HTTP 200 Ok this response conforms the unauthorized access.

Impact:

An attacker can view or manipulate other user's data. This cause data exposure and unauthorized access.

Vulnerability 2:

A2 – Cryptographic Failures (JWT Information Disclosure)

Sensitive user information like credentials is stored inside a JWT token. JWT payloads are only Base64 encoded and not encrypted, attackers can easily decode and read sensitive data.

Affected Component - JWT Authentication Token

Steps:

1. Login and capture the JWT token.
 2. Decode the token using base64 or jwt.io.
 3. Observe sensitive user details in the payload.

Result:

Decoded JWT payload reveals email, user ID, role, and password hash.

The screenshot shows a JSON Web Token (JWT) being analyzed on jwt.io. The token is valid and contains the following payload:

```

{
  "typ": "JWT",
  "alg": "RS256"
}

DECODED PAYLOAD
{
  "status": "success",
  "data": {
    "id": 23,
    "username": "",
    "email": "jifit7167901200b.com",
    "password": "25f9e794323b453885f5181f1b624d0b",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage": "/assets/public/images/uploads/default.svg",
    "totpSecret": "",
    "isActive": true,
    "createdAt": "2026-01-26 07:19:21.251 +00:00",
    "updatedAt": "2026-01-26 07:19:21.251 +00:00",
    "deletedAt": null
  }
},

```

Impact:

Sensitive information disclosure can lead to privacy violations and offline password attacks.

Vulnerability 3:

A3 – Injection (SQL Injection)

The login page does not properly validate user input, it allowing payload injection to bypass authentication.

Affected Endpoint- Post /rest/user/login

Steps:

1. Intercept login request.
2. Inject malicious payload in email/password field.
3. Forward the request.

Result:

Authentication was bypassed successfully using payload injection.

Impact:

Attacker can gain unauthorized access to user accounts.

Vulnerability 4:

A5 – Security Misconfiguration (Exposed FTP)

FTP directory was accessible without proper restrictions; it causes unauthorized file access.

Affected Component – FTP Service

Result:

FTP directory was accessible directly.

Impact:

Sensitive files can be download or modified by attackers.

Vulnerability 5:

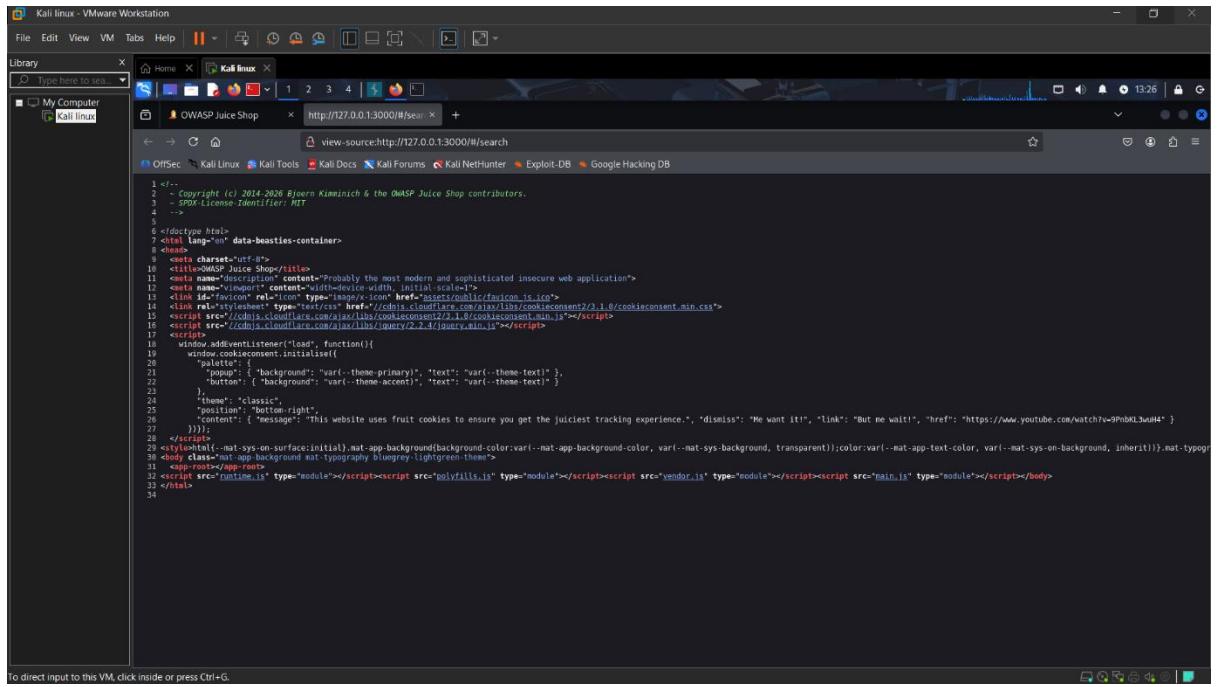
A6 – Vulnerable and Outdated Components

The Application uses jQuery v2.2.4 it contains known security vulnerabilities.

Affected Component – jQuery v2.2.4

Proof:

Version identified from page source code.



```
1 <!--
2 -- Copyright (c) 2014-2016 Bjoern Kaminich & the OWASP Juice Shop contributors.
3 SPDX-License-Identifier: MIT
4 --
5 <!doctype html>
6 <html lang="en" data-beauties-container>
7 <head>
8   <meta charset="utf-8">
9   <title>OWASP Juice Shop</title>
10  <meta name="description" content="Probably the most modern and sophisticated insecure web application">
11  <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
12  <link id="apple-touch-icon" type="image/x-icon" href="assets/public/favicon.ico">
13  <link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent/3.1.0/cookieconsent.min.css">
14  <script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent/3.1.0/cookieconsent.min.js"></script>
15  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
16  <script>
17    window.addEventListener("load", function(){
18      window.cookieconsent.initialise({
19        "palette": {
20          "popup": { "background": "#var(--theme-primary)", "text": "#var(--theme-text)" },
21          "button": { "background": "#var(--theme-accent)", "text": "#var(--theme-text)" }
22        },
23        "theme": "classic",
24        "position": "bottom-right",
25        "content": { "message": "This website uses fruit cookies to ensure you get the juiciest tracking experience.", "dismiss": "We want it!", "link": "But no wait!", "href": "https://www.youtube.com/watch?v=3PhbKL3wH4" }
26      });
27    });
28  </script>
29 <style>html{--mat-sys-on-surface:initial},mat-app-background{background-color:#var(--mat-app-background-color, var(--mat-sys-background, transparent));color:#var(--mat-app-text-color, var(--mat-sys-on-background, inherit))}.mat-typogr
30 body.class="mat-app-background mat-typography bluegrey-lightgreen-theme">
31 <script src="polyfills.js"></script>
32 <script src="runtime.js" type="module"></script><script src="polyfills.js" type="module"></script><script src="vendor.js" type="module"></script><script src="main.js" type="module"></script></body>
33 </html>
34
```

Impact:

Known vulnerabilities may allow XSS or client-side attacks.

Mitigation for Each Vulnerability:

A1 – Implement server-side authorization checks.

A2 – Do not store sensitive data in JWT payloads.

A3 – Use parameterized queries.

A5 – Restrict access with authentication.

A6 – Upgrade jQuery to latest stable version.