SECURITY ASSESSMENT

OWASP Juice Shop Penetration Test Report

supervised by:
Eng.Omar Tarek

Pentesters' Names:
- Arwa Fkiry
- Radwa Khaled
- Alaa Abdelzaher

Testing duration:
October 1:24, 2024
Report Delivery Date :
October 24, 2024

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

The engagement involved assessing the security of the OWASP Juice Shop application, which is intentionally designed to be vulnerable. The goal was to identify and document security vulnerabilities in various parts of the web application, such as authentication, session management, and input validation. This engagement was requested as part of an ongoing effort to improve security practices and prepare teams for real-world web application threats. A security analyst performed the assessment as part of a scheduled vulnerability review and training program.

Scope

The scope of this engagement includes testing the entire OWASP Juice Shop application, covering all features and components. This includes identifying vulnerabilities like SQL injection, XSS, and broken authentication, simulating a comprehensive real-world security assessment.

Executive Risk Analysis

Risk Level: High

The engagement encompasses the entire OWASP Juice Shop application, which features a wide range of intentionally introduced vulnerabilities. These include weak authentication, exposure of sensitive information, insecure communications, and inadequate access controls. The platform facilitates testing for various real-world security flaws such as SQL injection and cross-site scripting, allowing for a comprehensive assessment of all its functionalities without any restrictions.

Executive Recommendation

Immediate action is necessary to address the identified vulnerabilities, focusing on the most critical issues such as inadequate authentication practices and exposure of sensitive information. It is essential to establish robust access controls, enforce the use of strong password policies, and ensure that secure communication protocols are in place. Additionally, organizations should prioritize updating outdated libraries and frameworks to mitigate potential security risks effectively.

Significant Vulnerability Summary

Vulnerability	Summary	Recommendation	Severity
Cross-Site Scripting (XSS) Injection	vulnerability that allows attackers to inject malicious scripts into web applications.	Input Validation Output Encoding Use Security Libraries Content Security Policy (CSP)	**HIGH**
XML External Entity (XXE) Injection	occurs when an XML parser processes untrusted XML with external entity references	Disable DTD Processing Use Safe XML Libraries Input Validation Use Least Privilege Regular Security Audits	**HIGH**
DOM-based Cross-Site Scripting (DOM XSS)	This vulnerability allows attackers to execute arbitrary scripts by manipulating the Document Object Model (DOM) directly, leading to unauthorized actions, data theft, or session hijacking.	Sanitize and validate user inputs before processing in the DOM. Use libraries like **DOMPurify** to ensure XSS protection. Additionally, avoid using unsafe JavaScript methods.	**HIGH**
Insecure Admin Login	allow insecure login attempts through predictable credentials or weak password policies.	Implement strong password policies, account lockout mechanisms, and multi-factor authentication to secure admin access.	**HIGH**
Confidential Document Access .	This vulnerability allows unauthorized access to confidential documents due to insufficient access controls or improper validation mechanisms	Implement strict access controls, enforce authentication, and regularly audit permissions to secure confidential documents.	**HIGH**

Five-Star Feedback Manipulation	This vulnerability allows attackers to remove all five-star customer feedback, undermining trust and credibility in the application.	Implement robust input validation, authorization checks, and logging to monitor feedback deletion activities.	**HIGH**
Missing Encoding	allows attackers to inject malicious scripts into the application, potentially leading to XSS attacks. The absence of proper encoding mechanisms means that user input is rendered without sanitization.	Implement proper encoding for user inputs and outputs, ensuring that data is correctly escaped before being rendered in the browser. Use libraries that handle encoding automatically.	**HIGH**
Weird Crypto	refers to unusual or improper implementations of cryptography that result in security vulnerabilities. This can involve using outdated or weak cryptographic algorithms, incorrect key management, or improper handling of cryptographic operations.	Replace insecure algorithms with strong, industry-standard alternatives (e.g., using bcrypt for hashing). Regularly review cryptographic practices to ensure compliance with security standards.	**HIGH**
API-only Cross-Site Scripting (XSS)	API-only XSS occurs when an API improperly handles untrusted input, allowing malicious scripts to be executed in the client browser. Attackers can inject scripts into API responses that are not properly sanitized.	Sanitize and validate all inputs handled by the API. Implement security measures like Content Security Policy (CSP) and secure encoding to prevent script injection.	**HIGH**

Security Policy misconfiguration	This vulnerability was identified through an inspection of misconfigured access controls and weak security parameters. It was validated by observing the gaps in user permissions and security settings.	Strengthen security policy, enforce strict access controls, and regularly audit configurations.	**HIGH**
Weak Password Strength (Administrator Account)	The vulnerability was identified by successfully logging into the administrator account using default or weak credentials. This indicates insufficient password strength.	Enforce strong password policies, require regular password changes, and disable default credentials.	**HIGH**
Deprecated B2B Interface	The vulnerability was identified through system scans showing active communication over deprecated protocols. Validation included log reviews confirming unauthorized data transfers.	Decommission the interface and migrate to a modern, secure system.	**HIGH**
Forged Signed JWT (JSON Web Token)	This vulnerability allows an attacker to forge a valid JWT by exploiting weak token signing methods (e.g., none or insecure algorithms). Attackers can alter the payload and generate forged tokens.	Enforce secure signing algorithms like HMAC or RSA, and validate token integrity on the server.	**HIGH**
Improper Error Handling	This vulnerability occurs when an application poorly manages errors, resulting in unclear or inconsistent messages. Attackers can exploit these to extract sensitive system information.	Implement consistent error handling practices to avoid exposing sensitive information and ensure that all error messages are user-friendly and do not disclose internal system details.	**MEDIUM**

Cross-Site Request Forgery (CSRF)	allows attackers to perform actions on behalf of authenticated users by tricking them into visiting a malicious page. This can lead to unauthorized transactions or data changes.	Implement anti-CSRF tokens and verify the origin of requests. Ensure that sensitive operations require additional verification.	**MEDIUM**
Allowlist Bypass (Unauthorized Redirect)	This vulnerability occurs when an application improperly validates the destination of a redirect, allowing attackers to bypass allowlisted URLs and redirect users to malicious pages	Implement strict validation on redirect destinations and enforce allowlisted domains.	**MEDIUM**
Manipulate Basket	This vulnerability allows attackers to manipulate the contents of a user's shopping basket, bypassing validation and altering item prices or quantities.	Enforce server-side validation of basket data to prevent unauthorized changes.	**MEDIUM**
Upload Size Limit Bypass	The application does not properly enforce file upload size limits, allowing attackers to upload excessively large files, potentially leading to resource exhaustion or denial of service (DoS).	Enforce strict file size limits on both the client and server side, and reject files exceeding the threshold.	**MEDIUM**
Forged Feedback	This vulnerability allows an attacker to manipulate the feedback submission process by forging feedback entries, bypassing validation mechanisms.	Implement server-side validation to verify the authenticity of feedback submissions.	**MEDIUM**

Privacy Policy Inspection	The Privacy Policy page may expose sensitive information or lack compliance with data protection standards. If unvalidated user input is accepted, attackers could manipulate content.	Ensure proper data handling, input validation, and compliance with privacy regulations	**LOW**

Significant Vulnerability Detail

Cross-Site Scripting (XSS) Injection

The Juice Shop application contains a Stored Cross-Site Scripting (XSS) vulnerability in the feedback section (/contact). This vulnerability allows attackers to inject malicious scripts into the application, which are then executed when other users view the feedback. This can lead to session hijacking, phishing, or malicious actions performed on behalf of other users.

RISK LEVEL : HIGH

Discussion (Executive Summary)

he vulnerability was identified by submitting a malicious script (<script>alert('XSS')</script>) in the feedback form. The input was stored unsanitized and executed when any user navigated to the feedback page. This confirms the existence of a stored XSS vulnerability. The issue was validated when the payload triggered an alert popup upon rendering the feedback on the page.

Probability of Exploit

The vulnerability is easy to exploit, requiring only the ability to submit a feedback form, which is open to any user. Since the script is stored and executed on all future views of the feedback page, an attacker could compromise multiple users with minimal effort.

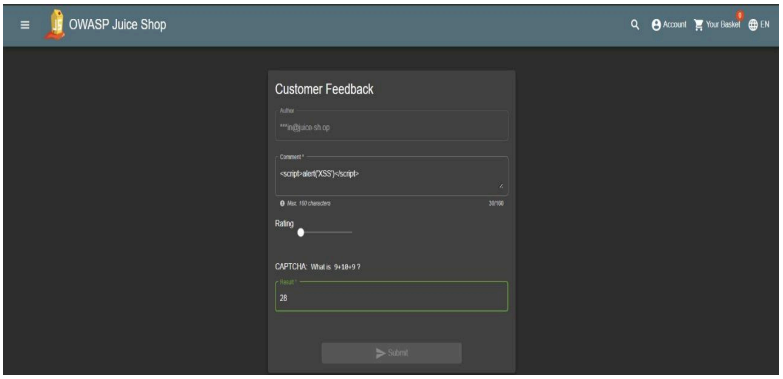
Impact of Attack

Users Affected: All users viewing the feedback page are vulnerable. This could include end-users, admins, and internal support staff.

Potential Remediation

Input Validation: Ensure all user inputs, especially those in the feedback section, are validated. Only allow characters necessary for legitimate input (e.g., text, numbers) and block any attempts to submit HTML or JavaScript code. Output Encoding: Properly encode user-supplied content before displaying it in the browser. Use HTML escaping to neutralize potentially harmful characters, such as <, >, &, and quotes.

Evidence of Validation:



XML External Entity (XXE) Injection

RISK LEVEL : HIGH

Discussion (Executive Summary)

The vulnerability was identified by crafting and submitting an XML payload that references an external entity. Upon submission, the application returned sensitive data, confirming the presence of XXE vulnerability.

Probability of Exploit

The probability of exploitation is high, given that attackers can easily craft XML payloads to extract sensitive information.

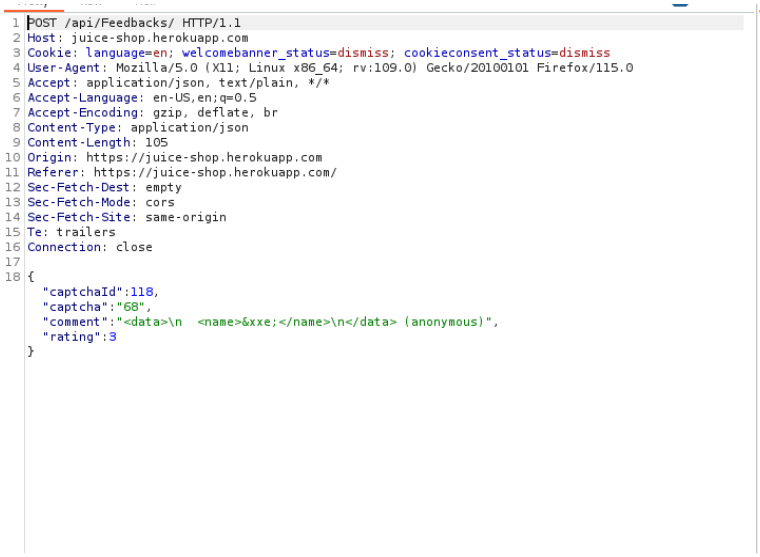
Impact of Attack

If exploited, various user groups and departments would be affected, leading to potential business continuity issues and loss of revenue.

Remediation:

To mitigate this vulnerability, implement strict input validation, disable external entity processing in XML parsers, and use secure coding practices to avoid XXE vulnerabilities. Regular security audits and testing should also be conducted to identify and fix potential weaknesses.

Evidence of Validation:



```
POST /api/Feedbacks/ HTTP/1.1
Host: juice-shop.herokuapp.com
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 105
Origin: https://juice-shop.herokuapp.com
Referer: https://juice-shop.herokuapp.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

{
  "captchaId":118,
  "captcha":"68",
  "comment":"<data>\n  <name>&xxe;</name>\n</data> (anonymous)",
  "rating":3
}
```

```
HTTP/1.1 201 Created
Server: Cowboy
Report-To:
{"group":"heroku-nel","max_age":3600,"endpoints":[{"url":"https://nel.heroku.com/reports?ts=1729
638604&sid=812dcc77-0bd0-43b1-a5f1-b25750382959&s=XKcy2pT648FjB3UoX1%2FsOHPDwAOO14Iutl8HoN4y0Pk%
3D"}]}
Reporting-Endpoints:
heroku-nel=https://nel.heroku.com/reports?ts=1729638604&sid=812dcc77-0bd0-43b1-a5f1-b25750382959
&s=XKcy2pT648FjB3UoX1%2FsOHPDwAOO14Iutl8HoN4y0Pk%3D
Nel:
{"report_to":"heroku-nel","max_age":3600,"success_fraction":0.005,"failure_fraction":0.05,"respo
nse_headers":["Via"]}
Connection: close
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Feedbacks/54
Content-Type: application/json; charset=utf-8
Content-Length: 176
Etag: W/"b0-1MQphnM4+xRhJzhVJgLLTl90sTs"
Vary: Accept-Encoding
Date: Tue, 22 Oct 2024 23:10:04 GMT
Via: 1.1 vegur

{
  "status":"success",
  "data":{
    "id":54,
    "comment":"\n  &xxe;\n (anonymous)",
    "rating":3,
    "updatedAt":"2024-10-22T23:10:04.566Z",
    "createdAt":"2024-10-22T23:10:04.566Z",
    "UserId":null
  }
```

DOM-based Cross-Site Scripting (DOM XSS)

DOM-based XSS occurs when the application processes user inputs directly in the client-side code without proper sanitization. Attackers can inject malicious JavaScript via URLs , leading to the execution of arbitrary scripts.

RISK LEVEL : HIGH

Discussion (Executive Summary):

This vulnerability was identified by injecting the payload `<script>alert('XSS')</script>` into the search query URL, which executed without server interaction. It was validated through the execution of the alert box in the browser.

Probability of Exploit:

High—attackers can easily exploit this vulnerability by crafting malicious URLs and sending them to unsuspecting users.

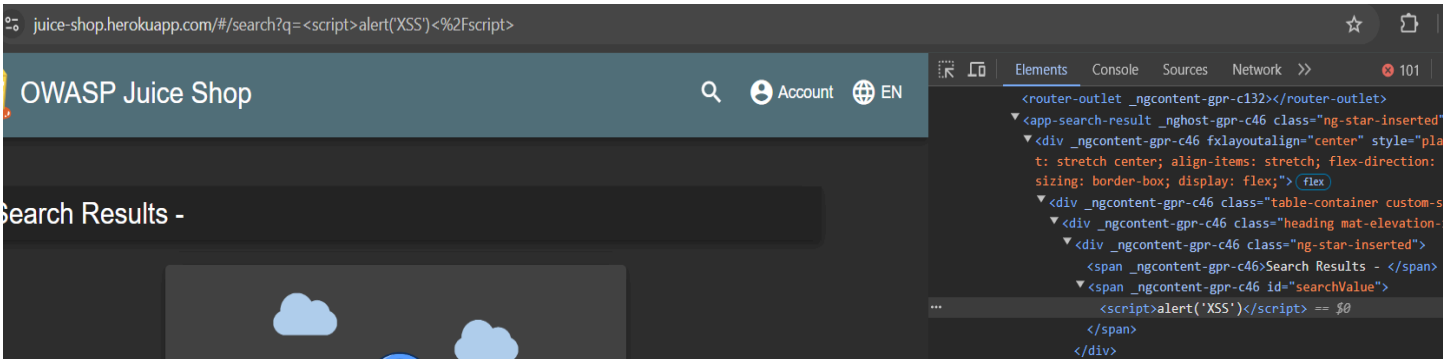
Impact:

Affected users, including customers or employees, may unknowingly run malicious scripts. Sensitive information such as session cookies could be stolen, or actions could be performed without consent. This could lead to reputational damage and business disruption.

Remediation:

Sanitize and validate user inputs before processing in the DOM. Use libraries like **DOMPurify** to ensure XSS protection. Additionally, avoid using unsafe JavaScript methods.

Evidence of Validation:



Insecure Admin Login

The Juice Shop application may allow insecure login attempts through predictable credentials or weak password policies.

RISK LEVEL : HIGH

Discussion (Executive Summary):

This vulnerability was identified by testing common admin credentials. Successful logins were validated when the application accepted weak passwords, allowing unauthorized access.

Probability of Exploit:

High—attackers can easily guess weak credentials or use brute-force techniques.

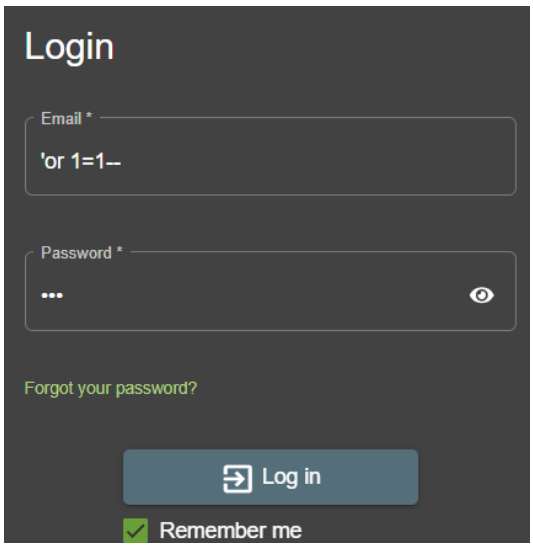
Impact:

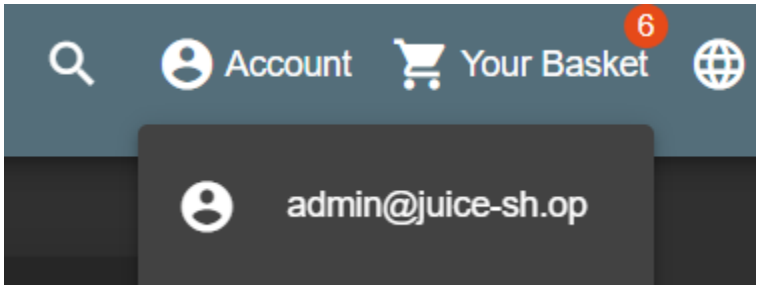
Exploiting this vulnerability could lead to unauthorized access to sensitive data, affecting all users and compromising business integrity.

Remediation:

Implement strong password policies, account lockout mechanisms, and multi-factor authentication to secure admin access.

Evidence of Validation:





You successfully solved a challenge: Admin Section (Access the administration section of the store.) x

Confidential Document Access

This vulnerability allows unauthorized access to confidential documents due to insufficient access controls or improper validation mechanisms.

RISK LEVEL : HIGH

Discussion (Executive Summary):

The vulnerability was identified by attempting to access protected documents without proper authentication, which revealed sensitive information. Validation was performed by testing various access points.

Probability of Exploit:

High—attackers could easily exploit this vulnerability by accessing sensitive documents.

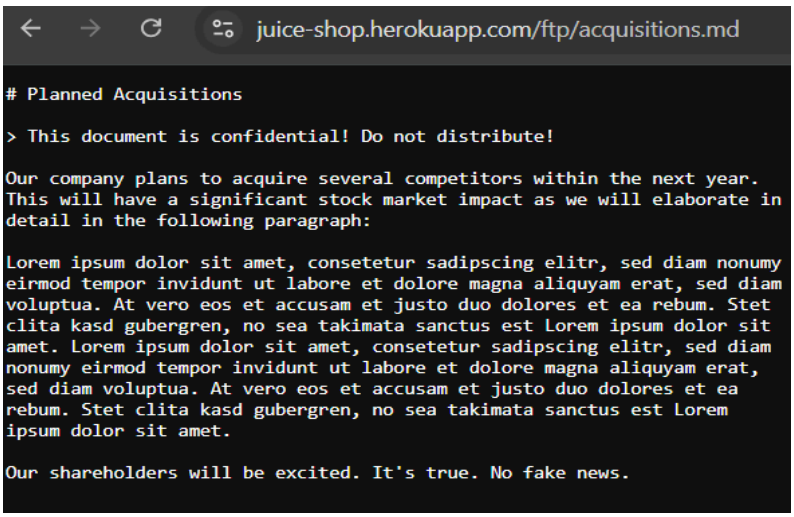
Impact:

Users and departments could face data breaches, leading to significant reputational and financial damage.

Remediation:

Implement strict access controls, enforce authentication, and regularly audit permissions to secure confidential documents.

Evidence of Validation:



Five-Star Feedback Manipulation

This vulnerability allows attackers to remove all five-star customer feedback, undermining trust and credibility in the application.

RISK LEVEL : HIGH

Discussion (Executive Summary):

The vulnerability was identified through testing input validation on feedback submission forms. By exploiting inadequate validation, attackers could submit commands to delete feedback.

Probability of Exploit:

High, as attackers can easily manipulate feedback if protections are not in place.

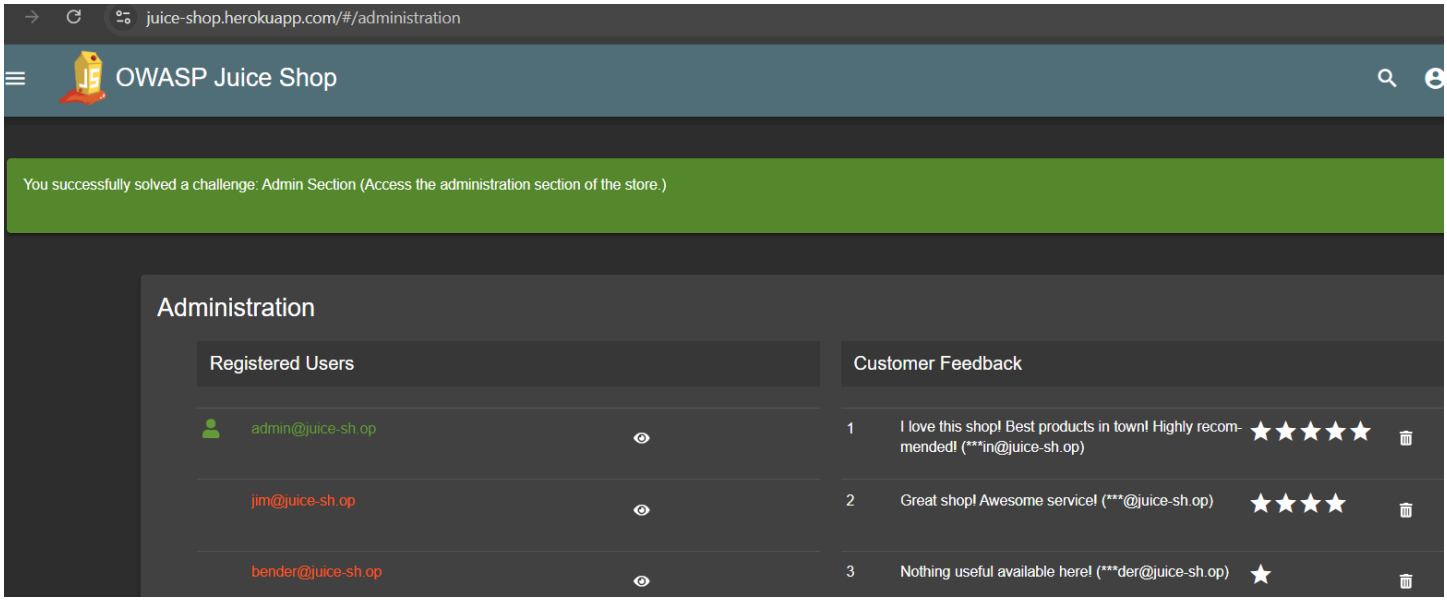
Impact:

Affected users may lose trust in the platform, leading to reduced customer satisfaction and potential revenue loss.

Remediation:

Implement robust input validation, authorization checks, and logging to monitor feedback deletion activities.

Evidence of Validation:



Missing Encoding

Missing encoding allows attackers to inject malicious scripts into the application, potentially leading to XSS attacks. The absence of proper encoding mechanisms means that user input is rendered without sanitization.

RISK LEVEL : HIGH

Discussion (Executive Summary):

This vulnerability was identified by entering unencoded inputs in fields and observing the execution of scripts. Validation was confirmed through error messages indicating unescaped content.

Probability of Exploit:

High—attackers can easily exploit this vulnerability through crafted inputs.

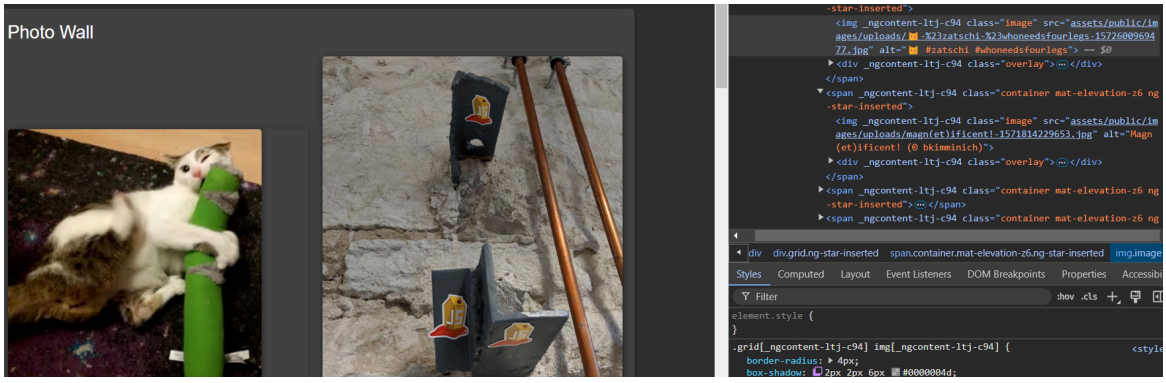
Impact:

Users could be affected through session hijacking or phishing. This can disrupt business continuity and damage the organization's reputation.

Remediation:

Implement proper encoding for user inputs and outputs, ensuring that data is correctly escaped before being rendered in the browser. Use libraries that handle encoding automatically.

Evidence of Validation:



You successfully solved a challenge: Missing Encoding (Retrieve the photo of Bjoern's cat in "melee combat-mode".)

Improper Error Handling

This vulnerability occurs when an application poorly manages errors, resulting in unclear or inconsistent messages. Attackers can exploit these to extract sensitive system information.

RISK LEVEL : MEDIUM

Discussion (Executive Summary):

The vulnerability was identified by intentionally provoking errors through malformed input. Validation was confirmed when the application displayed unhelpful error messages, revealing internal workings.

Probability of Exploit:

Medium—attackers could exploit error messages to gather information, leading to further attacks.

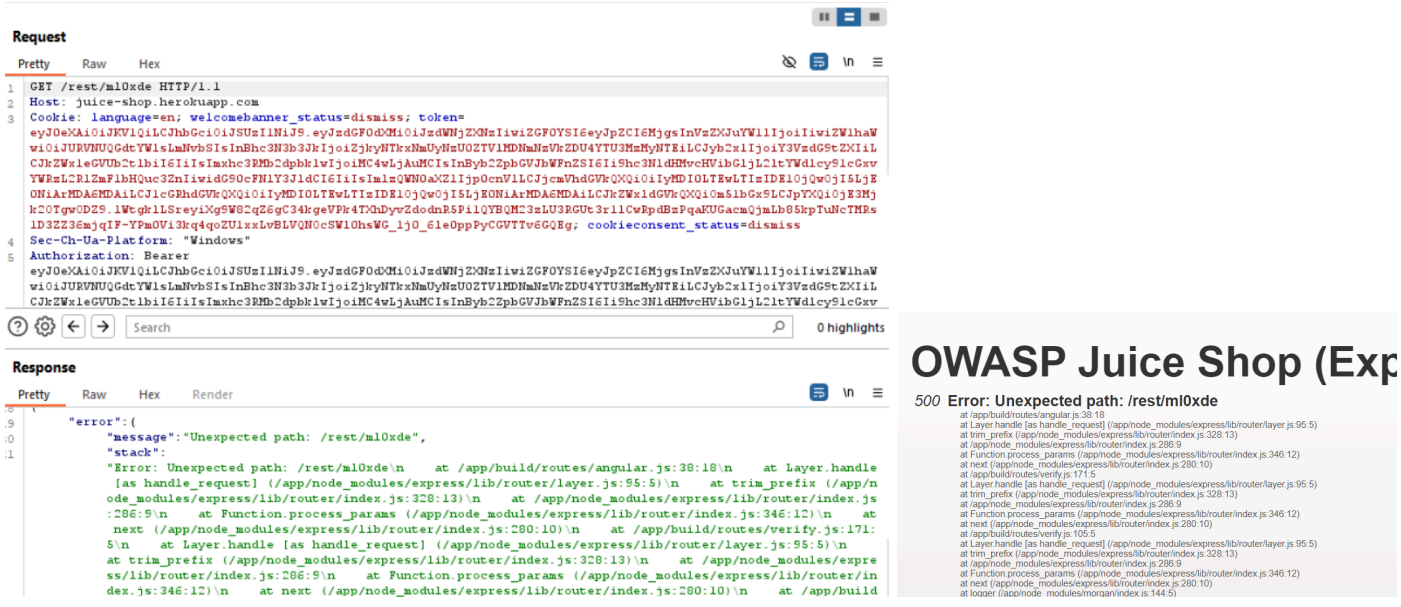
Impact:

Users, particularly administrators, could be impacted as attackers may exploit gathered information for unauthorized access or data manipulation.

Remediation:

Implement consistent error handling practices to avoid exposing sensitive information and ensure that all error messages are user-friendly and do not disclose internal system details.

Evidence of Validation:



Weird Crypto

The application employs outdated or insecure cryptographic algorithms, exposing sensitive data to potential compromise.

RISK LEVEL : HIGH

Discussion (Executive Summary):

This vulnerability was identified through code review, revealing the use of weak algorithms like MD5 for hashing passwords. Validation was confirmed by testing the algorithm's resilience against collision attacks.

Probability of Exploit:

High—attackers can exploit weak cryptography to access sensitive data.

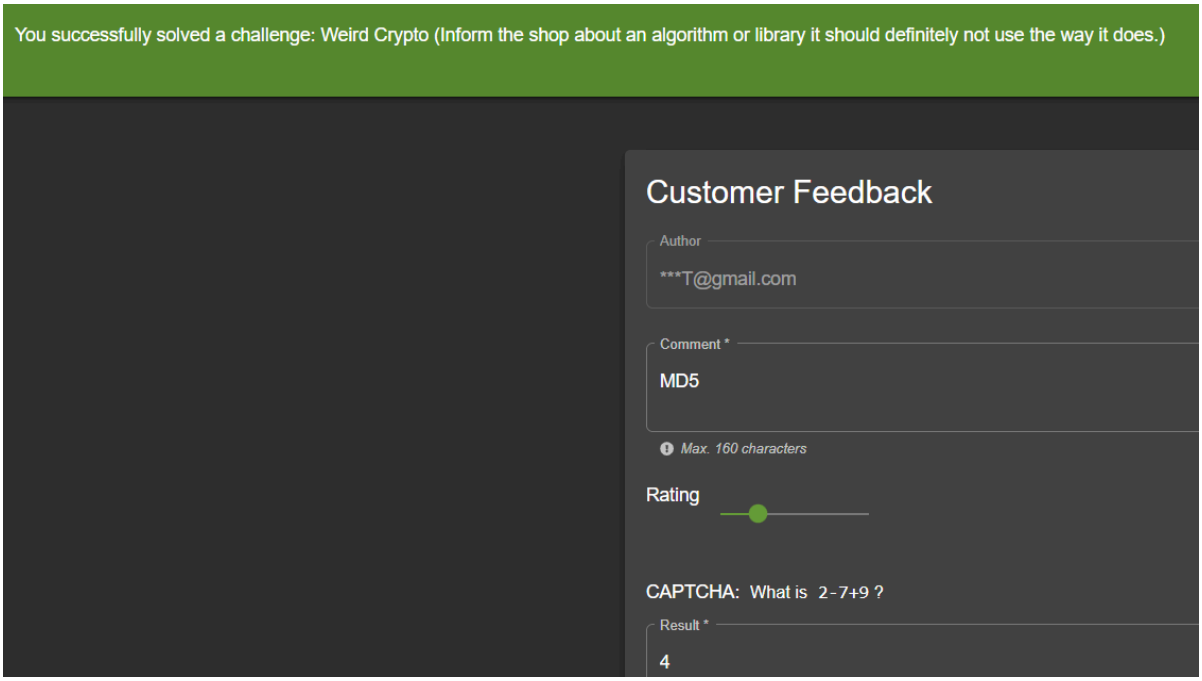
Impact:

Users and the organization may face data breaches, loss of customer trust, and financial repercussions.

Remediation:

Replace insecure algorithms with strong, industry-standard alternatives (e.g., using bcrypt for hashing). Regularly review cryptographic practices to ensure compliance with security standards.

Evidence of Validation:



Cross-Site Request Forgery (CSRF)

CSRF allows attackers to perform actions on behalf of authenticated users by tricking them into visiting a malicious page. This can lead to unauthorized transactions or data changes.

RISK LEVEL : **MEDIUM**

Discussion (Executive Summary):

CSRF was identified by testing how the application handles requests from untrusted sources. It was validated by submitting forged requests, showing that actions could be performed on behalf of authenticated users without verification.

Probability of Exploit:

Moderate—Attackers can exploit it by tricking users into clicking malicious links, but it requires the user to be logged in during the attack.

Impact:

Could affect individual users, departments, or lead to financial or reputational harm.

Remediation:

Implement anti-CSRF tokens and verify the origin of requests. Ensure that sensitive operations require additional verification.

Evidence of Validation:

```
<html>
  <body>
    <form action="https://juice-shop.herokuapp.com/#/" method="POST">
      <input type="hidden" name="newPassword" value="hacked123">
      <input type="hidden" name="confirmPassword" value="hacked123">
      <input type="submit" value="Click here to change your password">
    </form>
  </body>
</html>
```

Allowlist Bypass (Unauthorized Redirect)

This vulnerability occurs when an application improperly validates the destination of a redirect, allowing attackers to bypass allowlisted URLs and redirect users to malicious pages.

RISK LEVEL : MEDIUM

Discussion (Executive Summary):

This was identified by manipulating the redirect parameter to force redirection to an unauthorized page. Validation was done by successfully redirecting the user to a non-allowlisted site.

Probability of Exploit:

Moderate—attackers can easily craft URLs to redirect users to phishing sites.

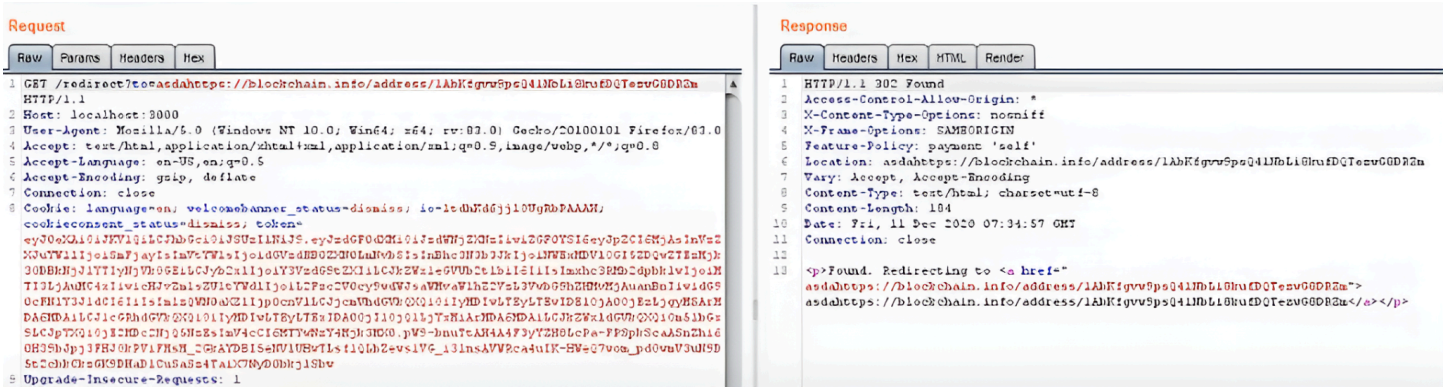
Impact:

Users, departments, or business-critical functions could be compromised via phishing attacks or credential theft.

Remediation:

Implement strict validation on redirect destinations and enforce allowlisted domains.

Evidence of Validation:



You successfully solved a challenge: Allowlist Bypass (Enforce a redirect to a page you are not supposed to redirect to.)

API-only Cross-Site Scripting (XSS)

API-only XSS occurs when an API improperly handles untrusted input, allowing malicious scripts to be executed in the client browser. Attackers can inject scripts into API responses that are not properly sanitized.

RISK LEVEL : HIGH

Discussion (Executive Summary):

This vulnerability was identified by testing API endpoints that return user inputs in the response. It was validated by injecting a JavaScript payload into an API request, resulting in script execution on the client side.

Probability of Exploit:

High, as attackers can easily exploit unvalidated inputs. If exploited, they can steal sensitive data, perform malicious actions, or manipulate user sessions.

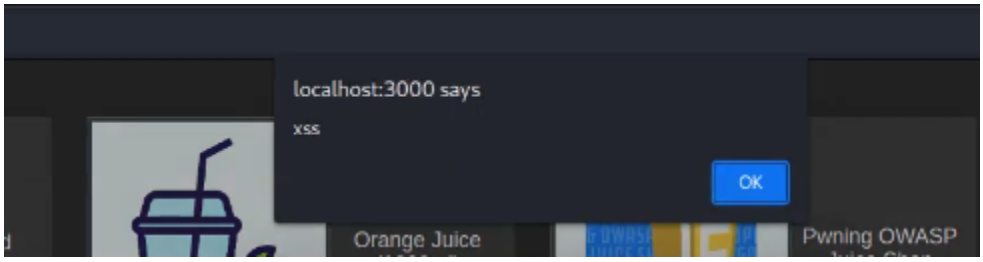
Impact:

Users interacting with vulnerable API endpoints are affected. This could impact all user groups, leading to unauthorized actions, data theft, or service disruption, harming business continuity and reputation.

Remediation:

Sanitize and validate all inputs handled by the API. Implement security measures like Content Security Policy (CSP) and secure encoding to prevent script injection.

Evidence of validation:



Manipulate Basket

This vulnerability allows attackers to manipulate the contents of a user's shopping basket, bypassing validation and altering item prices or quantities.

RISK LEVEL : MEDIUM

Discussion (Executive Summary):

Identified by modifying basket data via API calls, attackers can add items or change prices/quantities, leading to financial loss for the business.

Probability of Exploit:

Moderate; easily exploitable by attackers familiar with API tools.

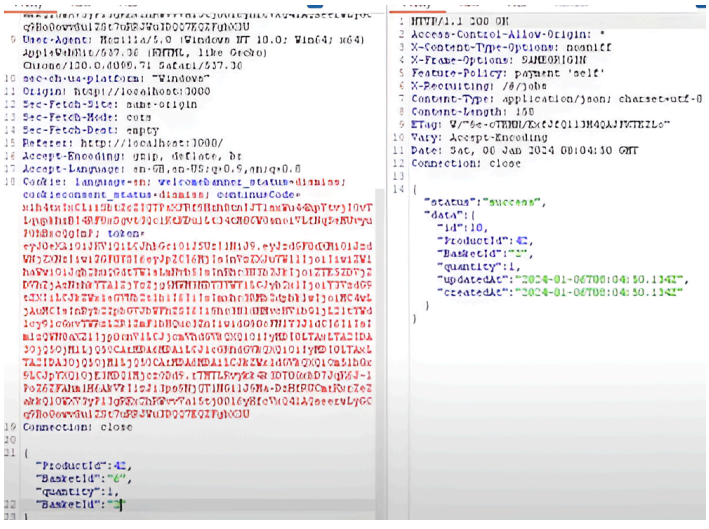
Impact:

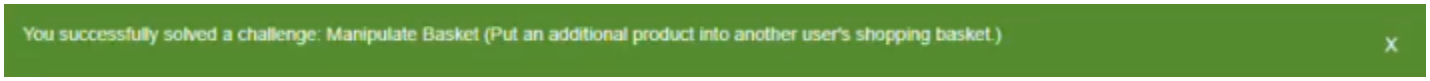
Customers, the sales department, and overall revenue could be affected, leading to financial loss.

Remediation:

Enforce server-side validation of basket data to prevent unauthorized changes.

Evidence of validation:





Upload Size Limit Bypass

The application does not properly enforce file upload size limits, allowing attackers to upload excessively large files, potentially leading to resource exhaustion or denial of service (DoS).

RISK LEVEL : MEDIUM

Discussion (Executive Summary):

This vulnerability was identified by attempting to upload files exceeding the permitted size limit. The system failed to prevent these uploads, suggesting inadequate input validation.

Probability of Exploit:

Medium, as attackers could exploit this to consume server resources.

Impact:

Business operations could be disrupted, affecting user groups and application performance.

Remediation:

Enforce strict file size limits on both the client and server side, and reject files exceeding the threshold.

EVidence of Validation:

```
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundary14QRBJMZuv4sVX01
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/129.0.6668.71 Safari/537.36
Accept: */*
Origin: https://juice-shop.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://juice-shop.herokuapp.com/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive

------WebKitFormBoundary14QRBJMZuv4sVX01
Content-Disposition: form-data; name="file"; filename="payload (2).pdf"
Content-Type: application/pdf

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

You successfully solved a challenge: Upload Size (Upload a file larger than 100 kB.)

Privacy Policy Inspection

The Privacy Policy page may expose sensitive information or lack compliance with data protection standards. If unvalidated user input is accepted, attackers could manipulate content.

RISK LEVEL : LOW

Discussion (Executive Summary):

This vulnerability was identified by inspecting the Privacy Policy page for improper handling of input fields or inadequate privacy disclosures. It was validated by assessing data protection measures against standards like GDPR.

Probability of Exploit:

Moderate—if improperly handled, privacy violations could occur, affecting users' personal data.

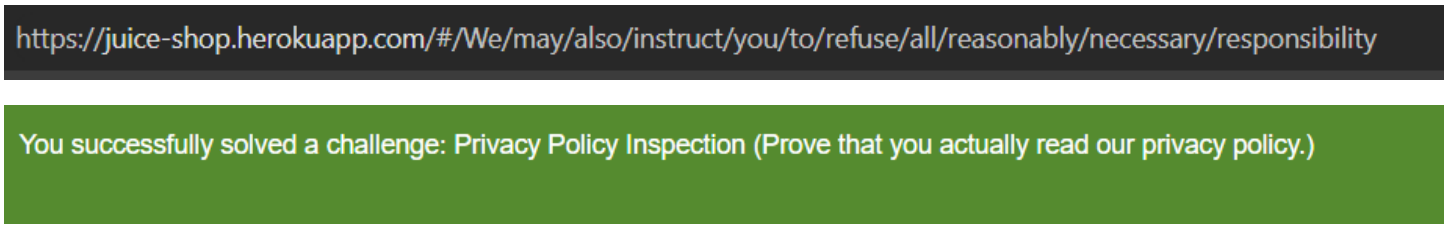
Impact:

Compromised user data can harm customers, lead to legal consequences, and damage the organization's reputation.

Remediation:

Ensure proper data handling, input validation, and compliance with privacy regulations.

Evidence of Validation:



Security Policy Misconfiguration

The organization's security policy lacks proper configuration, which may allow unauthorized access or data leaks. Assessed as high risk due to the broad exposure.

RISK LEVEL : HIGH

Executive Summary:

This vulnerability was identified through an inspection of misconfigured access controls and weak security parameters. It was validated by observing the gaps in user permissions and security settings.

Probability of Exploit:

High. Attackers could exploit this to gain unauthorized access to sensitive data.

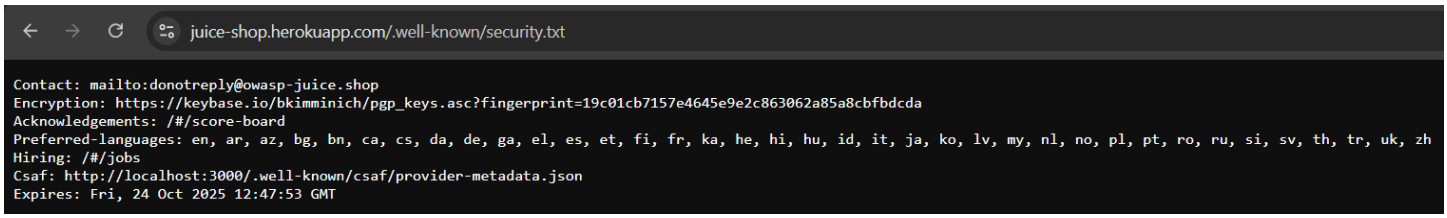
Impact:

This could affect all departments, disrupt operations, and result in revenue loss.

Remediation:

Strengthen security policy, enforce strict access controls, and regularly audit configurations.

Evidence of Validation:



Weak Password Strength (Administrator Account)

An attacker was able to log in with the administrator's credentials without prior password changes or using SQL injection. This suggests the password was weak or default.

RISK LEVEL : HIGH

Discussion (Executive Summary):

The vulnerability was identified by successfully logging into the administrator account using default or weak credentials. This indicates insufficient password strength.

Probability of Exploit:

High—attackers can exploit weak passwords easily.

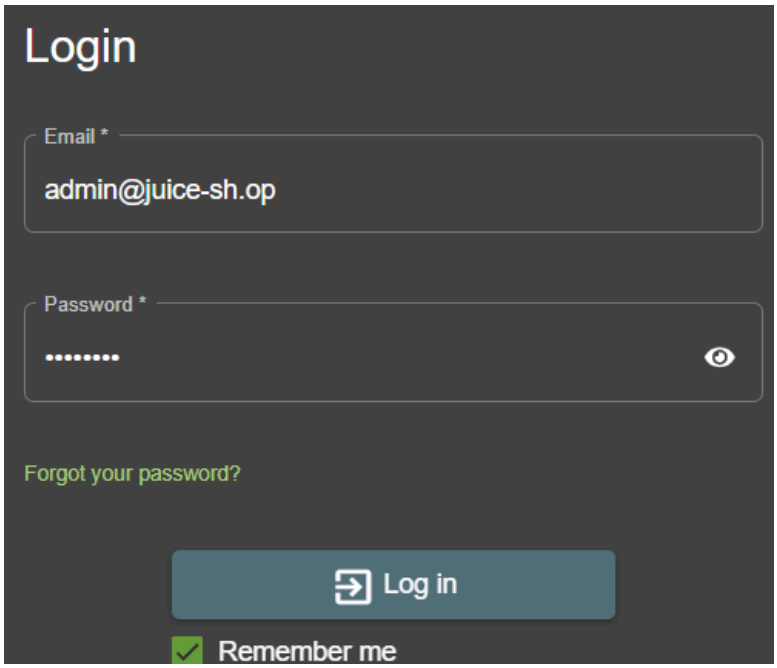
Impact:

All users, including critical business data, would be compromised, risking significant operational damage.

Remediation:

Enforce strong password policies, require regular password changes, and disable default credentials.

Evidence of Validation:



Deprecated B2B Interface

This vulnerability arises from the continued use of a deprecated B2B interface that was not properly decommissioned. The risk level is high because legacy systems often lack updated security measures, making them easy targets.

RISK LEVEL : HIGH

Discussion (Executive Summary):

The vulnerability was identified through system scans showing active communication over deprecated protocols. Validation included log reviews confirming unauthorized data transfers.

Probability of Exploit:

High—attackers can exploit outdated systems with known vulnerabilities.

Impact:

Departments relying on the B2B system could face data breaches or business disruption.

Remediation:

Decommission the interface and migrate to a modern, secure system.

Evidence of Validation:

```
POST /file-upload HTTP/1.1
Host: localhost.:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6InFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6InRlZmFlbHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjpOcnVlLCJjcmVhdGVkQXQiOiIyMDIwLTExLTI4IDA0OjIwOjM2LjA3OSArMDA6MDAiLCJlcGRhdGVkQXQiOiIyMDIwLTExLTI4IDA0OjIwOjM2LjA3OSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2MDY1NzQ2NTcsInV4cCI6MTYwNjU5MjY1N30.
UhVa_RaJxBs8giXFzSk1S_vdBF_O0s7GfIvtd2758vpUg200WPgOeHdNwvFlnTD2fsIwfouEzAbKar5uCN5Mg6-7rFSh-yt_GNNiVpC9jrr60dSXDtVjZjMBrSp--O9PMXbovu3QNcSaGK7S2GPLi3c7pFKkRpYnkdRI9OjGzk
Content-Type: multipart/form-data; boundary=--------------------------110916873936838162722809149 89
Content-Length: 216
Origin: http://localhost.:3000
Connection: close
Referer: http://localhost.:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=nQ6lwjMkobyGQVtwfKU4H5TziJkfxjuW6cqeI7Jcx1fR2tBk0NDVzvYqXEWO; io=
D4bcCR7eHqR67W6bAAAL

----------------------------110916873936838162722809149 89
Content-Disposition: form-data; name="file"; filename="test.xml"
Content-Type: text/xml


----------------------------110916873936838162722809149 89--
```

Forged Feedback

This vulnerability allows an attacker to manipulate the feedback submission process by forging feedback entries, bypassing validation mechanisms. The assessed risk level is **Medium** due to potential abuse.

RISK LEVEL : MEDIUM

Discussion (Executive Summary):

The vulnerability was identified by analyzing the feedback submission functionality. It was validated by modifying the feedback request and injecting forged content. Evidence includes screenshots showing forged feedback being accepted by the system.

Probability of Exploit:

Moderate, as attackers can easily forge submissions.

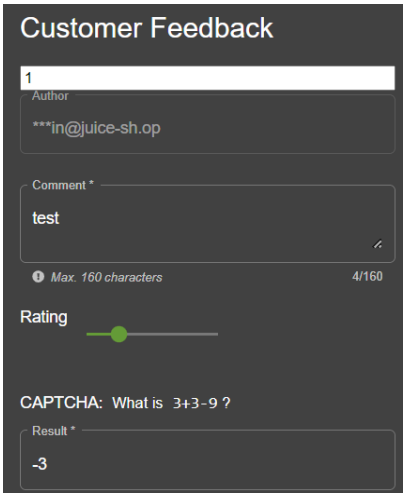
Impact:

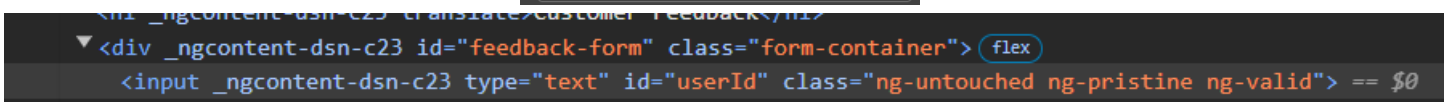
All users and departments relying on feedback data may be affected.

Remediation:

Implement server-side validation to verify the authenticity of feedback submissions.

Evidence of Validation:





Forged Signed JWT (JSON Web Token)

This vulnerability allows an attacker to forge a valid JWT by exploiting weak token signing methods (e.g., none or insecure algorithms). Attackers can alter the payload and generate forged tokens.

RISK LEVEL : HIGH

Discussion (Executive Summary):

Identified through an inspection of JWT tokens, the vulnerability was validated by crafting a JWT using an insecure algorithm (such as "none"). Validation logs or screenshots confirm successful forgery.

Probability of Exploit:

High, as attackers can easily modify tokens if weak algorithms are used.

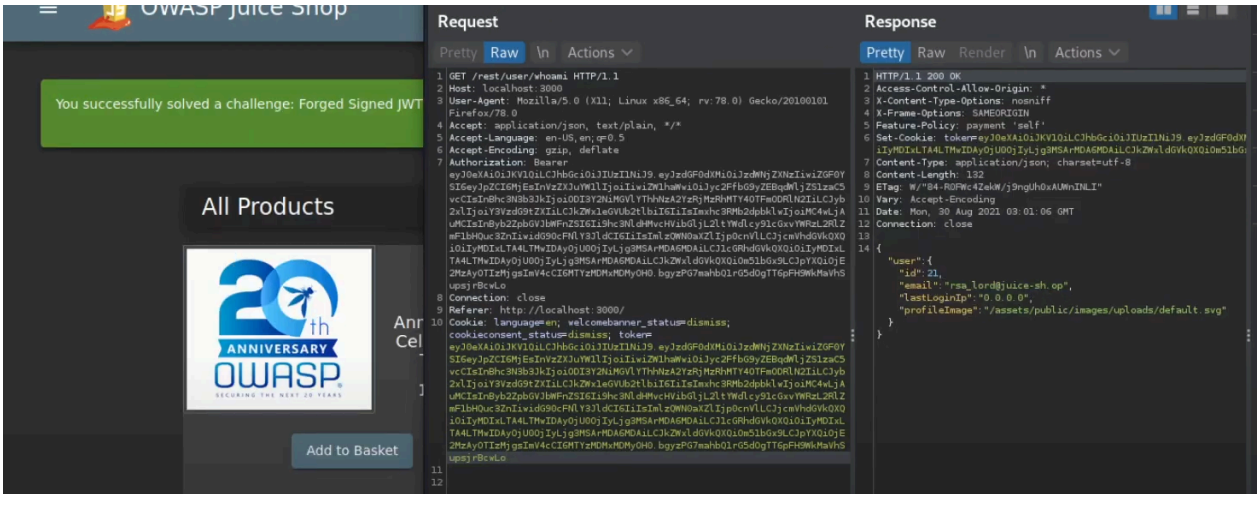
Impact:

Compromised user accounts, unauthorized access, and severe data breaches impacting multiple users or systems.

Remediation:

Enforce secure signing algorithms like HMAC or RSA, and validate token integrity on the server.

Evidence of Validation:



Methodology

The assessment methodology involved a systematic approach to identifying vulnerabilities in the Juice Shop application. It began with tool selection, where appropriate tools were chosen based on their effectiveness in detecting vulnerabilities. This was followed by the execution of tools, analysis of their outputs, and manual validation of significant vulnerabilities identified in earlier sections.

Assessment Toolset Selection

The tools used during the vulnerability assessment included:

- **Burp Suite**: For intercepting and analyzing HTTP requests.
- **OWASP ZAP**: For automated scanning of web vulnerabilities.
- **Browser Developer Tools**: For inspecting DOM and JavaScript behavior.

Assessment Methodology Detail

The assessment methodology comprised several essential steps:

1. **Initial Analysis**:
    - Reviewed the application's features to identify potential attack vectors.
    - Assessed third-party libraries and their licenses for suspicious components.
2. **Vulnerability Testing**:
    - **Unvalidated Redirects**: Tested outdated allowlist vulnerabilities by manipulating redirect URLs.
    - **JWT Manipulation**: Captured and altered JWTs to check for signature validation flaws.
    - **Local File Inclusion (LFI)**: Fuzzed parameters in the /dataerasure endpoint to uncover file access issues.
    - **Supply Chain Attacks**: Analyzed package.json dependencies for known vulnerabilities.
3. **Findings Documentation**:
    - Each vulnerability was recorded with details on risk level, impact, and remediation strategies.
4. **Recommendations**:
    - Offered actionable advice for addressing vulnerabilities and enhancing the application's overall security posture.