SECURITY ASSESSMENT

Juice Shop Vulnerabilities Report

Submitted to: DEPI

Security Analyst:
Khaled Muhammad Ibrahim Eldesoki
Hussein Kamal El-din
Fathy Magdy Elnady
Ameen Ahmed Refaat

Date of Testing: 20 October 2024

Date of Report Delivery: 24 October 2024

Table of Contents

Contents

SECURITY ENGAGEMENT SUMMARY 2

ENGAGEMENT OVERVIEW 2

SCOPE 2

EXECUTIVE RISK ANALYSIS 2

EXECUTIVE RECOMMENDATION 2

SIGNIFICANT VULNERABILITY SUMMARY 2

High Risk Vulnerabilities 2

Medium Risk Vulnerabilities 2

Low Risk Vulnerabilities 2

SIGNIFICANT VULNERABILITY DETAIL 3

SQL INJECTION IN LOGIN FORM 3

BROKEN AUTHENTICATION (BRUTEFORCE ADMINISTRATOR ACCOUNT) 4

BUSINESS LOGIC FLAW IN CHECKOUT (BUYING WITHOUT PAYING) 5

FORGED FEEDBACK SUBMISSION (IDOR) 6

CHANGE REVIEWS OF OTHER USERS 7

VIEW ANOTHER USER'S SHOPPING BASKET 8

DOM-BASED CROSS-SITE SCRIPTING (XSS) IN SEARCH FUNCTIONALITY 9

ACCESS TO /FTP DIRECTORY AND BACKUP FILES 10

WEAK PASSWORDS IN HIGH PRIVILEGED ACCOUNTS(WEAK PASSWORD POLICY) 11

CHANGE PASSWORD WITHOUT KNOWING CURRENT PASSWORD 12

METHODOLOGY 13

ASSESSMENT TOOLSET SELECTION 13

ASSESSMENT METHODOLOGY DETAIL 13

Security Engagement Summary

Significant Vulnerability Summary

High Risk Vulnerabilities

- **SQL Injection in Login Form**: The login functionality is vulnerable to SQL injection, allowing attackers to manipulate database queries and gain unauthorized access to user accounts.
- **Broken Authentication**: The application exhibits weaknesses that could be exploited through SQL injection or brute-force attacks on the administrator password, potentially leading to unauthorized administrative access.
- **Broken Access Control** (Access to admin panel via view main.js code)
- **Business Logic Flaw** in Checkout (Buying Without Paying)

Medium Risk Vulnerabilities

- **DOM-based XSS in Search Function**: The search functionality is susceptible to DOM-based cross-site scripting (XSS), enabling attackers to execute malicious scripts in users' browsers, which could compromise user data.
- **Unauthorized Access to FTP Directory**: Inadequate access controls have led to unauthorized access to the FTP directory, exposing sensitive files and application resources.
- **Weak Passwords** for High Privileged Accounts (Weak Password Policy)
- **Forged Feedback Submission (IDOR)**: The application allows for forged feedback submissions, which could lead to misinformation and damage the reputation of the service.
- **Change Reviews** of Other Users (IDOR)
- **View Another User's Shopping Basket**: The application permits users to view another user's shopping basket, which poses a minor privacy risk but does not significantly threaten data integrity or security.
-

Low Risk Vulnerabilities

- **Change Password** Without Knowing Current Password

Significant Vulnerability Detail

Each vulnerability is detailed below, with an executive summary of its nature, evidence of validation, impact, and recommended remediation.

SQL Injection in Login Form

Risk Level: High (9.0)

Vulnerability Type: Injection (OWASP A1)

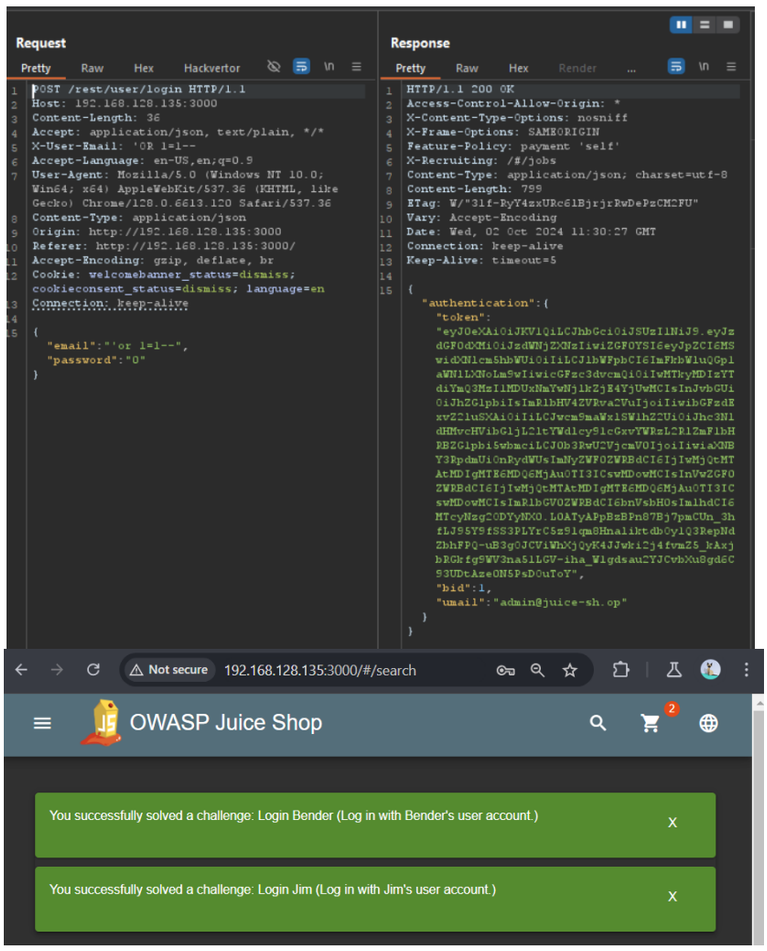
CWE Reference: CWE-89

- **Vulnerability Detail:**

  The SQL injection vulnerability was identified in the login form, allowing an attacker to bypass authentication mechanisms. During testing, a malicious SQL query was injected, which allowed unauthorized access to user accounts by manipulating the backend SQL database.

  The vulnerability was confirmed when we successfully logged in as users "bender" and "jim" by injecting SQL code into the username field, exploiting improper input sanitization. This allowed full account takeover without password knowledge

- **Evidence of Validation:**



.

- **Probability of Exploit:**
  This vulnerability is highly exploitable due to the lack of prepared statements and parameterized queries. Publicly available tools like SQLMap can automate this attack with ease, increasing the likelihood of compromise.
- **Impact:**
  If exploited, this vulnerability could allow attackers to access user accounts, including administrative accounts. This would lead to data leakage, unauthorized access to sensitive information, and potential full-system compromise. Such attacks can severely damage user trust, affect business continuity, and result in regulatory non-compliance (e.g., GDPR violations).
- **Affected Entities:**
  All application users, including system administrators, will be impacted by unauthorized access.
- **Potential Remediation:**
  Implement parameterized queries and input validation on all user-supplied data. Conduct thorough security reviews of all input-handling mechanisms and apply Web Application Firewall (WAF) rules to detect and block SQL injection attempts.

Broken Authentication (Bruteforce Administrator Account)

Risk Level: High (8.7)

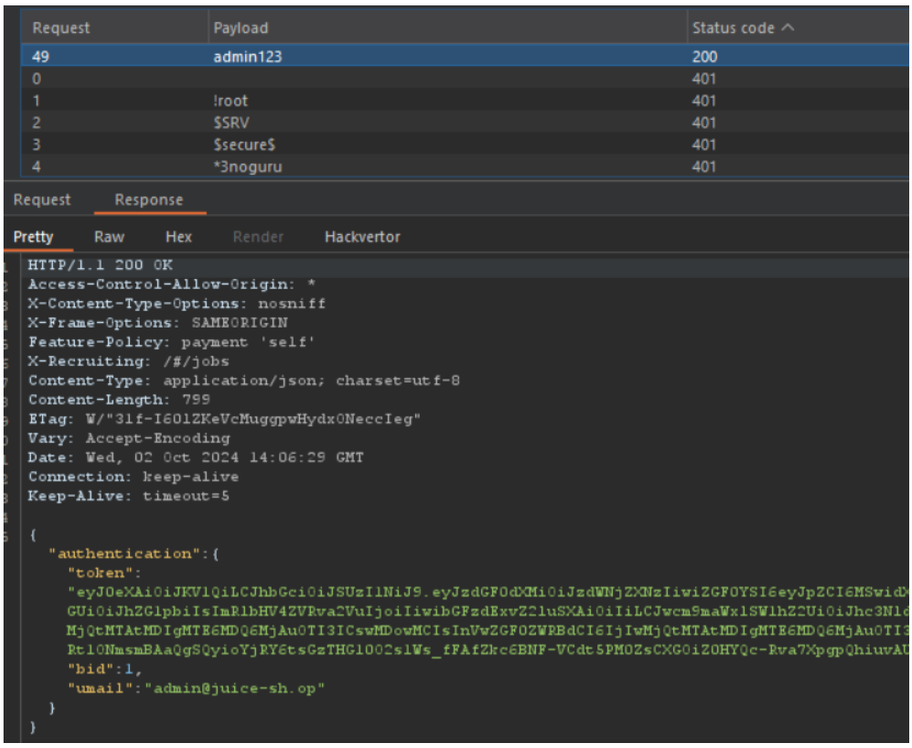
Vulnerability Type: Broken Authentication (OWASP A2)

CWE Reference: CWE-287

- **Vulnerability Detail:**

  The authentication system for the Administrator account is vulnerable to brute-force attacks due to the absence of security controls like rate-limiting, account lockout policies, or CAPTCHA mechanisms. During the assessment, we successfully brute-forced the administrator password using a standard wordlist, exploiting the lack of countermeasures.

- **Evidence of Validation:**



- **Probability of Exploit:**

  High. Without account lockouts or CAPTCHA, an attacker can repeatedly attempt login attempts, increasing the probability of successful brute-force attacks.

- **Impact:**

  A successful brute-force attack would grant full administrative privileges, allowing an attacker to alter critical system settings, access sensitive user data, and control the overall application infrastructure. Such a breach would have a catastrophic impact on business operations, regulatory compliance, and customer trust.

- **Affected Entities:**

  All administrative functions and users. The entire system's integrity could be jeopardized.

- **Potential Remediation:**

  Implement multi-factor authentication (MFA) for all administrator-level accounts, enforce CAPTCHA mechanisms, set strong password policies, and implement rate-limiting and account lockouts after a certain number of failed login attempts.

Business Logic Flaw in Checkout (Buying Without Paying)

Risk Level: High (8.5)

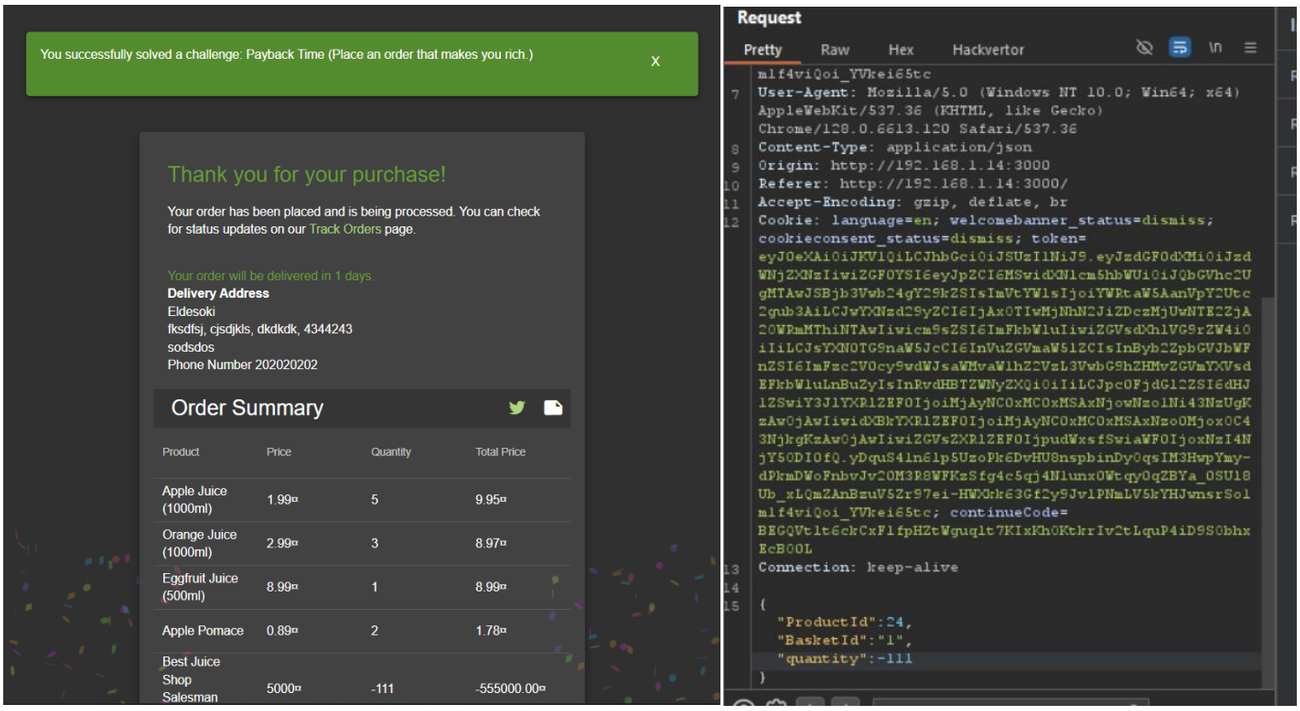
Vulnerability Type: Business Logic Vulnerability (OWASP A4)

CWE Reference: CWE-840

- **Vulnerability Detail:**

  The checkout system contains a business logic flaw that allows users to set negative quantities for products during the purchase process. By altering the quantity in the request, attackers can bypass payment entirely or generate refunds, leading to financial loss for the business. This was validated by changing the product quantity to -111, resulting in a total cost of 0 or a refund instead of payment.

- **Evidence of Validation:**



- **Probability of Exploit:**

  High. The attack can be easily executed by intercepting and modifying requests with tools like Burp Suite, making it highly exploitable.

- **Impact:**

  This vulnerability could lead to severe financial losses, as attackers could effectively "purchase" items without paying or generate fraudulent refunds. If exploited widely, this could severely impact the business's revenue and reputation.

- **Affected Entities:**

  The business and potentially its customers, as inventory and financial records would be compromised.

- **Potential Remediation:**

  Implement server-side validation to prevent negative quantities and ensure that all business logic processes are thoroughly validated to prevent manipulation. Validate all incoming requests for integrity on the server side.

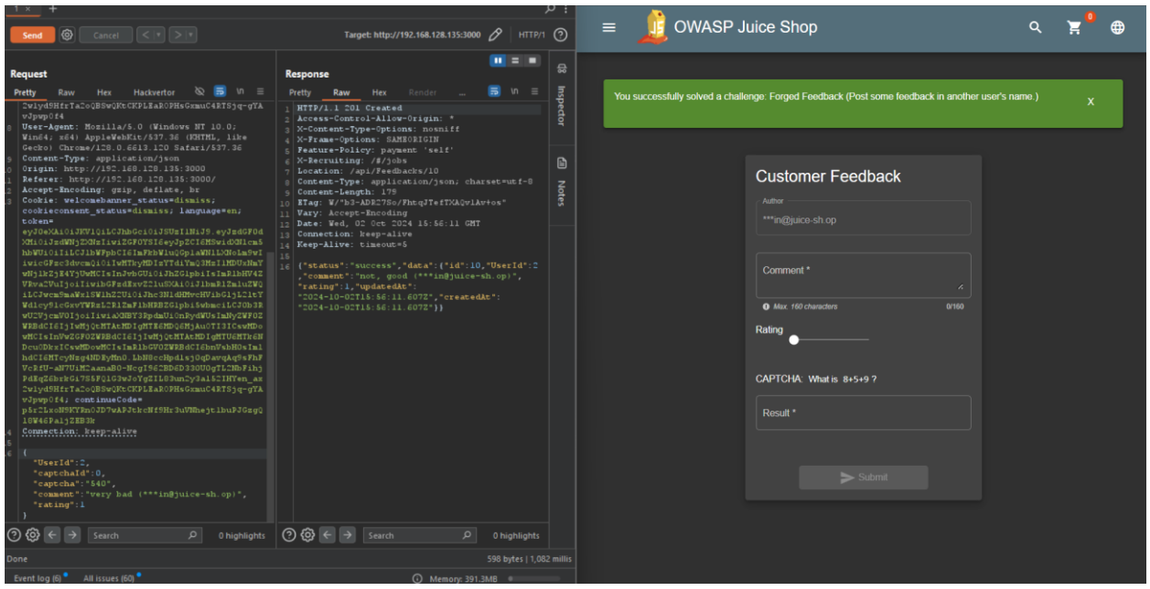
Forged Feedback Submission (IDOR)

Risk Level: Medium (6.0)

Vulnerability Type: Insecure Direct Object Reference (IDOR) (OWASP A5)

CWE Reference: CWE-639

- **Vulnerability Detail:**
  The feedback submission system is vulnerable to IDOR. By altering the userid in the POST request, attackers can submit feedback on behalf of other users without needing their session. This allows feedback to be posted under any user's name, bypassing authentication mechanisms.
- **Evidence of Validation:**



- **Probability of Exploit:**
  Medium. While attackers would need to know or guess valid user IDs, the lack of session checks makes this vulnerability exploitable, especially if user IDs are sequential.
- **Impact:**
  This allows attackers to manipulate the feedback system, potentially damaging user trust and the platform's reputation. If abused, this could lead to fraudulent reviews or malicious content being posted under legitimate users' names.
- **Affected Entities:**
  Users and the platform's reputation.
- **Potential Remediation:**
  Implement strict session and access control checks to ensure that the userid in the feedback form corresponds to the authenticated user. Do not allow modification of the userid in the request.

Change Reviews of Other Users

Risk Level: Medium (6.0)

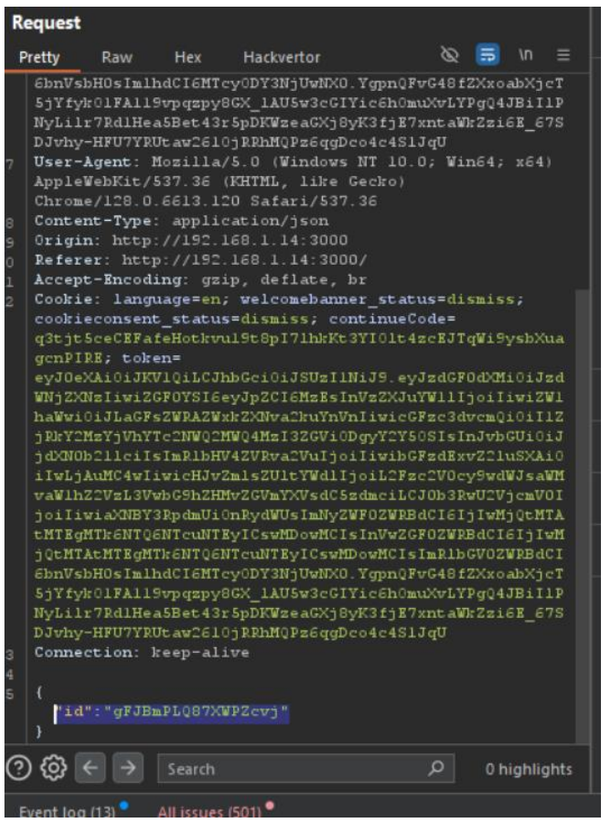
Vulnerability Type: Insecure Direct Object Reference (IDOR) (OWASP A5)

CWE Reference: CWE-639

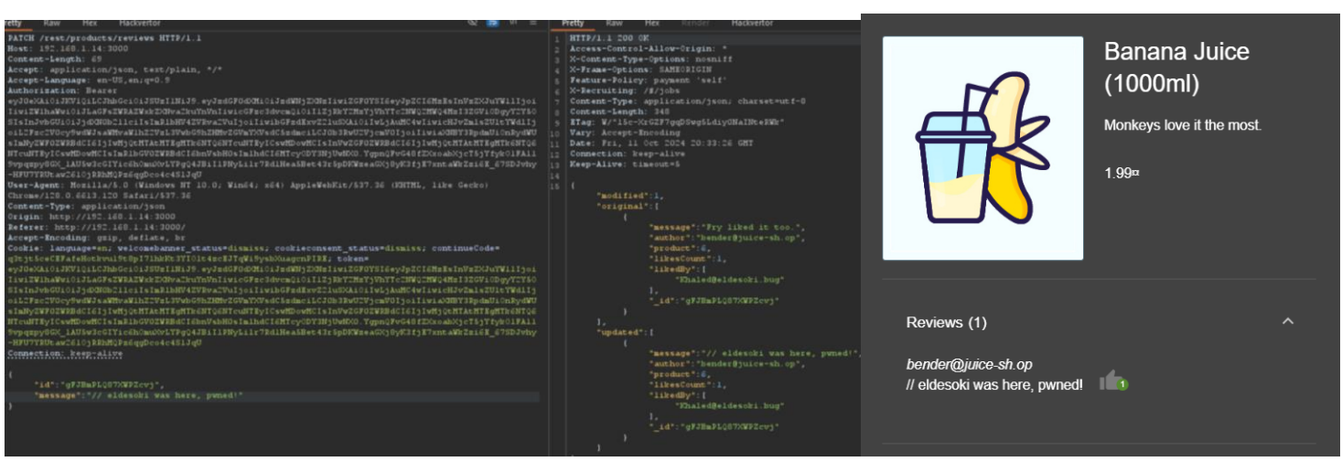
Vulnerability Detail:

- The review system allowed attackers to modify other users' reviews by manipulating the reviewid in the request. This flaw enabled attackers to change the content of reviews or add likes/dislikes without authorization.
- **Example**: Using Burp Suite, we intercepted the HTTP request containing the review's ID (reviewid). By changing this ID, we were able to edit or like/dislike another user's review.

- **Evidence of Validation**:



Put massage and id in patch request



- **Impact**:
  - This vulnerability compromises the integrity of the review system, as attackers can tamper with user feedback, leading to mistrust and potential manipulation of product or service ratings.
- **Potential Remediation**:
  - Implement strict validation to ensure that users can only modify reviews they have created. Also, ensure that reviewid is securely tied to the user session and cannot be arbitrarily modified.

View Another User’s Shopping Basket

- **Risk Level**: Medium (6.2)
- **Vulnerability Type**: Insecure Direct Object Reference (IDOR) (OWASP A5)
- **CWE Reference**: CWE-639
- **Vulnerability Detail**:
    - The application allowed attackers to view another user’s shopping basket by manipulating the basketid in the URL or API request. By altering the basket number, an attacker could access the contents of another user’s shopping basket, exposing sensitive information such as product details, pricing, and quantities.
    - **Example**: Intercepting a request to GET /rest/basket/1, we changed the basket ID from 1 to 7, successfully accessing another user's basket.
- **Evidence of Validation**:
    - Successfully accessed another user’s shopping basket by modifying the basket ID in the request.
- **Impact**:
    - Attackers can access sensitive user data such as items in the shopping basket, prices, and quantities, leading to privacy violations or further exploitation (e.g., buying items from another user's basket).
- **Potential Remediation**:
    - Implement access control mechanisms to ensure that users can only access their own basket. Use session-based validation to associate the basket ID with the authenticated user.

DOM-based Cross-Site Scripting (XSS) in Search Functionality

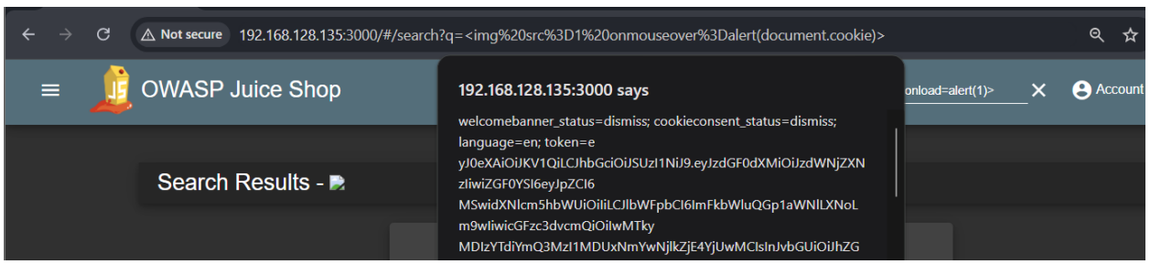
Risk Level: Medium (6.5)

Vulnerability Type: Cross-Site Scripting (OWASP A7)

CWE Reference: CWE-79

Vulnerability Detail:

- The search functionality is vulnerable to DOM-based XSS, allowing an attacker to inject and execute malicious JavaScript in the victim's browser. By altering the search query, an attacker can control the content rendered on the page, which can be used to steal session cookies, execute unauthorized actions on behalf of the user, or redirect victims to malicious sites.
- **Evidence of Validation:**



Probability of Exploit:

Medium. The exploit depends on user interaction, but it can be effectively leveraged through phishing attacks or social engineering to trick users into visiting a crafted URL.

- **Impact:**
  If exploited, this vulnerability could allow attackers to impersonate users, steal sensitive session information, or carry out malicious actions within the victim's browser session.
- **Affected Entities:**
  Users interacting with the search feature, particularly those with elevated permissions such as administrators.
- **Potential Remediation:**
  Ensure that all user input is properly sanitized and escaped before being rendered. Use security libraries like DOMPurify to sanitize inputs in JavaScript. Implement a Content Security Policy (CSP) to mitigate the impact of XSS attacks.

Access to /ftp Directory and Backup Files

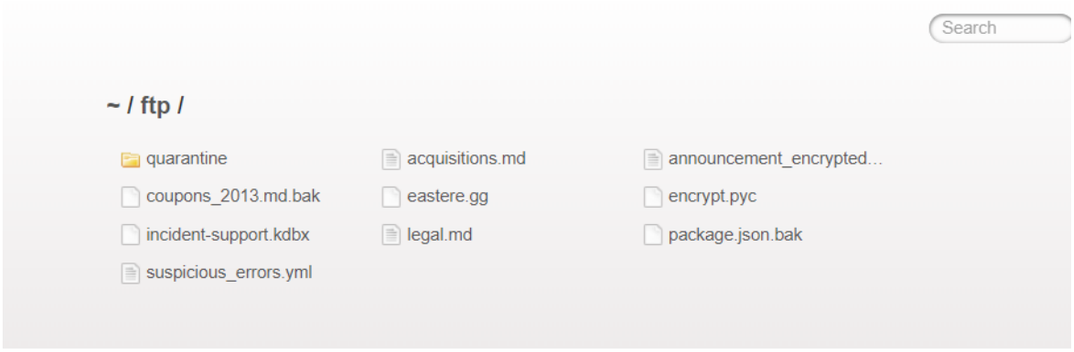
Risk Level: Medium (6.3)

Vulnerability Type: Improper Access Control (OWASP A5)

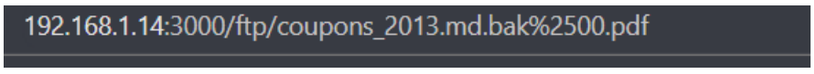
CWE Reference: CWE-284

Vulnerability Detail:

- The /ftp directory is publicly accessible and allows users to view and download backup files without authentication. This vulnerability was discovered by directly modifying the URL to access the FTP directory (http://localhost:3000/ftp) and successfully browsing the contents, which included sensitive backup files.
- **Evidence of Validation:**



And download it using null byte



- **Probability of Exploit:**
  Medium. While attackers would need to know the directory exists, the lack of authentication means anyone with access to the URL could easily download sensitive files.
- **Impact:**
  Unauthorized access to backup files can lead to the exposure of sensitive information, including user credentials, configuration files, and business-critical data. An attacker could restore these backups to analyze and potentially exploit the system further.
- **Affected Entities:**
  All users and any sensitive data stored in the backup files.
- **Potential Remediation:**
  Restrict access to the /ftp directory via proper authentication mechanisms and ensure backup files are stored in a secure, non-public location. Enforce least-privilege access for all system directories.

Weak Passwords in High Privileged Accounts(Weak Password Policy)

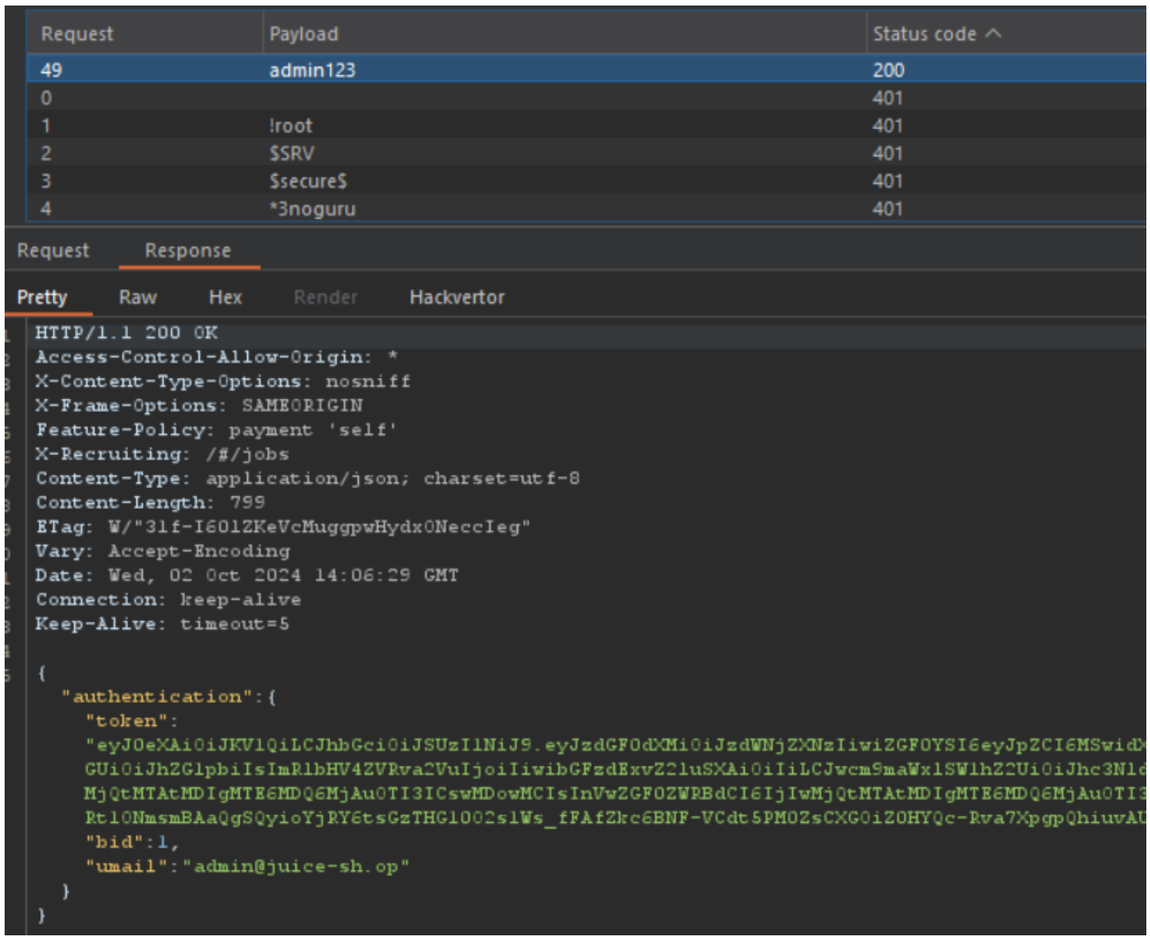
Risk Level: Medium (5.7)

Vulnerability Type: Weak Password Management (OWASP A2)

CWE Reference: CWE-521

Vulnerability Detail:

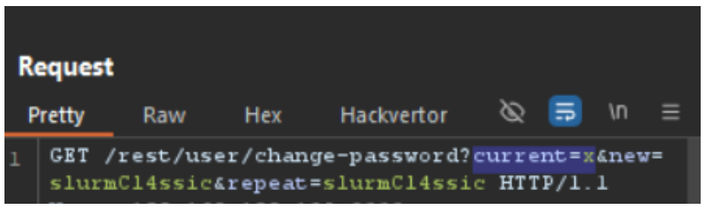
- High-privilege accounts, including administrator and root-level users, were found to use weak or common passwords. This was verified by successfully brute-forcing several high-privileged accounts using a basic wordlist, indicating inadequate password complexity policies.
- **Evidence of Validation:**



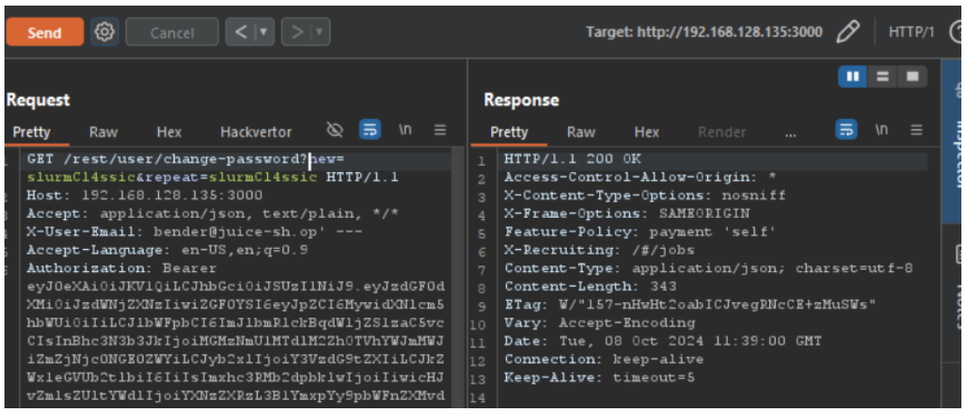
- **Probability of Exploit:**
  Medium. With access to a common wordlist and automated tools, an attacker could easily compromise privileged accounts.
- **Impact:**
  Compromise of high-privileged accounts would result in full control over the system, allowing attackers to modify configurations, access sensitive data, and even take over the application.
- **Affected Entities:**
  All administrators and high-privilege users.
- **Potential Remediation:**
  Enforce a strong password policy that requires the use of complex passwords, including length, special characters, and a combination of upper- and lower-case letters. Additionally, implement multi-factor authentication (MFA) to mitigate the risk.

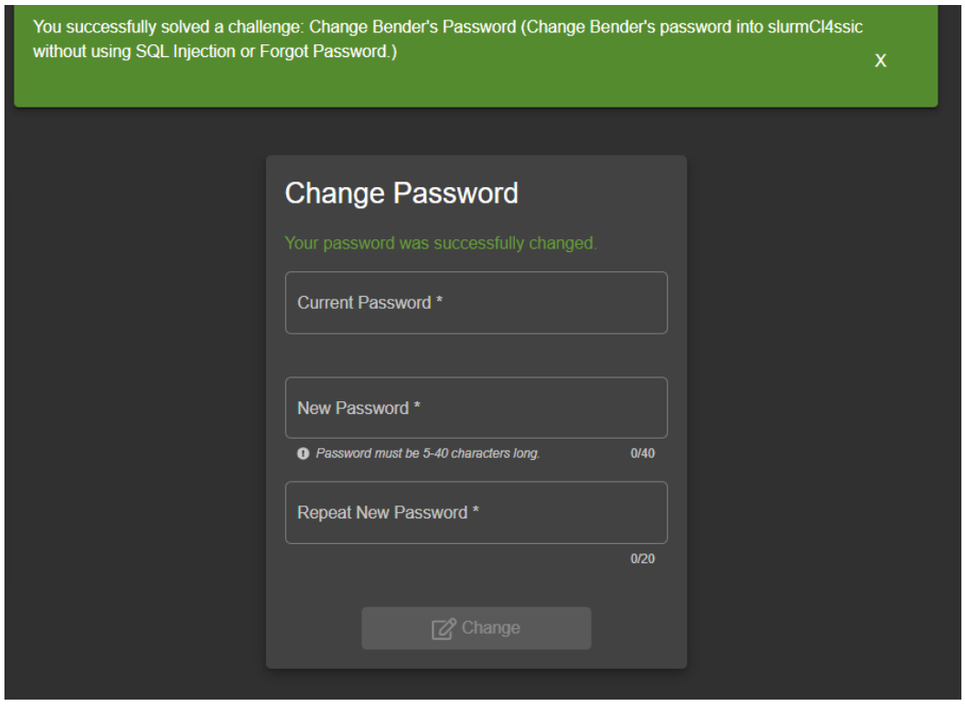
Change Password Without Knowing Current Password

- **Risk Level**: Low (3.8)
- **Vulnerability Type**: Improper Authentication (OWASP A2)
- **CWE Reference**: CWE-287
- **Vulnerability Detail**:
    - The system allowed users to change their passwords without entering the current password for verification. This lack of validation posed a security risk, as attackers could bypass authentication controls and change passwords for certain accounts without authorization.
    - **Example**: By removing the current_password field from the password change request, we were able to update the password for users like Jim, John, and Emma without knowing their existing password.
- **Evidence of Validation**:



remove current attribute form request





- **Impact**:
    - Attackers could reset passwords and lock legitimate users out of their accounts. This could also result in unauthorized access if attackers gained control of the account through password resets.
- **Potential Remediation**:
    - Enforce a policy requiring users to input their current password when changing it. Additionally, implement multi-factor authentication (MFA) to strengthen security around account modifications.

Methodology

Assessment Toolset Selection

The following tools were used during this assessment:
- **OWASP ZAP**: Used to identify injection and XSS vulnerabilities.
- **Burp Suite**: Used for manual testing and verifying vulnerabilities.

Assessment Methodology Detail

Preliminary Evaluation:

- Analyzed the application's features and identified potential attack vectors, focusing on user authentication processes and data management.
- Reviewed third-party libraries and their licenses to detect any outdated or suspicious components, ensuring alignment with security best practices.

Vulnerability Testing:

- **SQL Injection**: Utilized automated tools, such as SQLMap, to examine user input fields for SQL injection vulnerabilities, manipulating queries to evaluate database interactions.
- **Authentication Weaknesses**: Performed brute-force testing on the administrator login page with Burp Suite to uncover deficiencies in authentication controls.
- **Business Logic Vulnerability**: Investigated the checkout functionality to detect flaws that allowed transactions to be completed without payment by intercepting and altering HTTP requests.
- **Cross-Site Scripting (XSS)**: Leveraged OWASP ZAP to probe for XSS vulnerabilities in the search feature, testing various payloads to determine if user inputs could execute unauthorized scripts.
- **Insecure Direct Object Reference (IDOR)**: Examined the feedback submission system by manipulating user identifiers in requests to verify unauthorized access to restricted resources.

Findings Documentation:

- Documented each identified vulnerability comprehensively, including risk level (high, medium, low), potential impacts on the application, and suggested remediation measures to mitigate risks.

Proposed Recommendations:

- Offered actionable recommendations for addressing identified vulnerabilities, including:
    - Implementing parameterized queries and prepared statements to prevent SQL injection attacks.
    - Enhancing authentication methods with multi-factor authentication (MFA).
    - Validating user inputs to mitigate XSS and IDOR vulnerabilities.
    - Regularly updating and reviewing third-party dependencies to resolve known vulnerabilities.