The penetration testing of the OWASP Juice Shop application

Made with Gamma

Team Members

Noha Essam Muhammed

Karin Alaa gamel

Joseph khairy mansour

Moamen Abdelhakim Ali

Ethar Muhammed Nasr

Made with Gamma

1. Executive Summary

Penetration Test Report: OWASP Juice Shop
Prepared by: Team members / DEPI
Date: 20/10/2024
Target Application: OWASP Juice Shop

During the penetration testing of the OWASP Juice Shop application, multiple security vulnerabilities were identified, including SQL injection, improper access control in various login functionalities, unauthorized access to the Admin Section, exploitation of hidden features, direct access to the Score Board, View Basket, Five-Star Feedback, Privacy Policy page, a security misconfiguration related to error handling, and broken authentication due to weak password strength. These vulnerabilities allow attackers to manipulate data, retrieve sensitive information, bypass authentication, and gain unauthorized access to application features, posing significant risks to the application's data integrity and security.

Made with Gamma

2. Scope of Testing

- **Target Application**: OWASP Juice Shop
- **Tested URLs**: /rest/products/search?q=, admin login endpoint, Admin Section, Score Board, View Basket, Five-Star Feedback, Privacy Policy, error handling mechanisms, broken authentication scenarios, hidden login functionalities, and hidden special offers.
- **Vulnerability Types**: SQL Injection, Authentication Bypass, Hidden Functionality Exploitation, Unauthorized Access, Security Misconfiguration, Broken Authentication

Made with Gamma

3. Findings

1. **SQL Injection**
2. **Miscellaneous**
3. **Broken Authentication**
4. **Security Misconfiguration**
5. **Broken Access Control**
6. **Improper Input Validation**

Made with Gamma

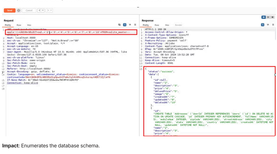
SQL Injection

1. SQL Injection Vulnerability in Product Search (Database Schema Extraction)

- **Description:** A SQL injection vulnerability in the product search functionality allows attackers to enumerate the database schema.
- **Impact:** Allows an attacker to enumerate the database schema and potentially manipulate database queries.
- **CVSS Score:** 8.2 (High)

Poc

SQL Injection Vulnerability in Product Search (Database Schema Extraction)



Impact: Enumerates the database schema.

2. SQL Injection Vulnerability in Product Search (User Credentials Extraction)

- **Description**: This vulnerability enables attackers to retrieve sensitive user data, including usernames and hashed passwords.
- **Impact**: Exposes sensitive user information, increasing the risk of unauthorized account access and data breaches.
- **CVSS Score**: 9.1 (Critical)

PoC

- **SQL Injection Vulnerability in Product Search (User Credentials Extraction)**

- **Impact**: Extracts user credentials.

3. Improper Access Control in Admin Login (Login Admin Vulnerability)

- **Description**: Improper access control allows unauthorized users to log in with administrative privileges.
- **Impact**: Allows attackers to gain administrative privileges, posing a significant threat to the application's security.
- **CVSS Score**: 8.5 (High)

PoC

.3. Improper Access Control in Admin Login (Login Admin Vulnerability)

- **Poc:**

- **Impact**: Allows unauthorized admin access.

4. Hidden Functionality Vulnerability: "Christmas Special"

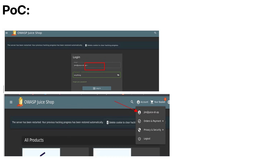
- **Description**: A hidden feature accessible via a specific URL grants unauthorized access to restricted offers.
- **Impact**: Unauthorized access to restricted offers can lead to financial exploitation or unintended user benefits.
- **CVSS Score**: 6.0 (Medium)

PoC

5/6 Authentication Bypass Vulnerabilities: "login jim" and "login bender"

- **Description**: Weak default credentials for the "jim" and "bender" accounts allow attackers to log in without strong authentication.
- **Impact**: Leads to easy exploitation by attackers, risking data theft and unauthorized actions.
- **CVSS Score**: 7.5 (High)

PoC:



Mitigation Recommendations

1. **Use Parameterized Queries:** (Prepared Statements) Ensure that any database queries are constructed using parameterized queries or prepared statements, which separate SQL code from user input. This way, user input is never directly concatenated into a query string, making it impossible for an attacker to inject malicious SQL code. Example in Python: `python Copy code cursor.execute("SELECT * FROM users WHERE username = %s AND password = %s", (username, password))` Prepared statements are supported in almost all programming languages.
2. **Implement ORM (Object-Relational Mapping) Frameworks:** Use ORM frameworks like SQLAlchemy (for Python), Hibernate (for Java), or Entity Framework (for .NET) to interact with the database. ORMs abstract away raw SQL and provide mechanisms to automatically escape dangerous characters in user input.
3. **Input Validation:** Validate and sanitize all inputs, including usernames, passwords, and other fields that interact with the database. Ensure the input only accepts the expected format (e.g., no special characters for usernames). For example, enforce a strict validation rule that usernames only contain alphanumeric characters.
4. **Stored Procedures:** Use stored procedures in the database to handle login and other sensitive queries. Stored procedures are precompiled, meaning user inputs are handled separately from the SQL code.
5. **Escape Special Characters:** If your application must use dynamic SQL queries, ensure you escape dangerous characters like `'`, `--`, and `;`, which are commonly used in SQL injection attacks.
6. **Use a Web Application Firewall (WAF):** A WAF can help detect and block common SQL injection attempts. Many modern WAFs have specific rules for blocking malicious injection attempts.
7. **Limit Database Privileges:** Ensure the database account used by the application has the least privileges necessary. For example: The account should not have administrative privileges or the ability to modify schema (e.g., no DROP TABLE or ALTER TABLE rights). Ideally, use role-based access control (RBAC) to limit what the application can do.
8. **Error Handling:** Ensure that detailed database errors are not shown to users. A generic error message should be displayed instead. Detailed errors can reveal sensitive information about the structure of the database, making it easier for attackers to craft injection payloads.
9. **Implement Strong Authentication:** Use multi-factor authentication (MFA) to strengthen admin login security. Ensure that the admin account uses a strong, complex password, and consider account lockout mechanisms to prevent brute-force attacks.
10. **Database Activity Monitoring:** Monitor the database for unusual queries or login attempts. Use logging and alerting mechanisms to detect potential injection attacks in real-time.

Methodology:

1. SQL Injection Vulnerability in Product Search (Database Schema Extraction)

Impact: Attackers can enumerate tables, columns, and potentially extract data.

Methodology:

1. **Reconnaissance**: Identify the product search input field or URL parameter where SQL queries are executed.
2. **Testing Input:** Test for SQL injection using basic payloads like:
   - `OR 1=1`
   - `OR '1'='1'`
3. **Blind SQL Injection:** Use payloads to infer schema information. For example:
   - **Boolean-based SQL Injection:** `' AND 1=1 --` or `' AND 1=2 --`
   - **Time-based SQL Injection:** `OR SLEEP(5) --`
4. **Schema Enumeration:** Once an SQL injection is confirmed, use enumeration payloads such as:
   - `UNION SELECT table_name FROM information_schema.tables --`
   - `UNION SELECT column_name FROM information_schema.columns WHERE table_name='users' --`

2. SQL Injection Vulnerability in Product Search (User Credentials Extraction)

Impact: Attackers can extract sensitive data, including usernames and hashed passwords.

Methodology:

1. **SQL Injection Confirmation:** Perform similar testing as in Vulnerability 1 to confirm an injection point.
2. **Union-Based Injection:** Use union-based SQL injection to retrieve specific data:
   - For instance, `UNION SELECT username, password FROM users --`
3. **Extract Hashed Passwords:** Once usernames and hashed passwords are retrieved, perform offline attacks (e.g., dictionary or rainbow table attacks).

3. Improper Access Control in Admin Login (Login Admin Vulnerability)

Impact: Attackers can gain unauthorized administrative access.

Methodology:

1. **Access Control Testing:** Analyze URL endpoints and authentication mechanisms for misconfigurations.
2. **Privilege Escalation:** Attempt to manipulate access by modifying cookies, session variables, or URL parameters.
3. **Authentication Bypass Techniques:** Test using various methods like insecure direct object reference (IDOR), forced browsing, or tampering JWT tokens if applicable.

4. Hidden Functionality Vulnerability: "Christmas Special"

Impact: Unrestricted URL access to hidden functionalities or offers.

Methodology:

1. **Fuzzing and URL Discovery:** Scan for hidden directories or endpoints using tools like **DirBuster** or **Burp Suite Intruder**.
2. **Test Privileged URLs:** Attempt to access hidden functionality using discovered URLs and analyze server responses.
3. **Access Validation:** Check if unauthorized users can access restricted offers.

5 & 6. Authentication Bypass Vulnerabilities: "login jim" and "login bender"

Impact: Weak default credentials allow unauthorized access.

Methodology:

1. **Enumerate Default Credentials:** Test using common or weak default credentials.
2. **Password Brute-Forcing:** Use a password list to brute-force weak accounts.
3. **Check for Role Privileges:** Verify if these accounts have roles granting higher-level privileges.

Made with Gemini

Broken Authentication

1. Unauthorized Brength Vulnerability in Administrator Login

- **Description:** Brute force authentication attacks exploit weak password policies, leveraging automated tools to systematically guess credentials.
- **Impact:** This vulnerability allows attackers to bypass authentication mechanisms, potentially compromising user accounts and sensitive data.
- **Severity:** High
- **CVSS Score:** 7.5 (High)

PoC

1. Go to the login page

2. Enter an OTP and click on Verify button and capture the request

3. Intruder payload

Recommendations

- Implement Multi-Factor Authentication (MFA)
  - Enforce multi-factor authentication (MFA) for administrator accounts, including a combination of passwords and one-time passwords (OTP) or biometrics.
  - Use trusted authentication apps (e.g., Google Authenticator) or hardware tokens for enhanced security.
- Implement Rate Limiting and Account Lockout Mechanisms
  - Limit the number of login attempts per IP address and per account to prevent brute force attacks.
  - Temporarily lock accounts after a specified number of failed attempts (e.g., 5 attempts within 15 minutes).
- Secure Session Management
  - Ensure session tokens are randomly generated and transmitted securely over HTTPS.
  - Implement session timeouts and invalidate sessions after logout or inactivity.
- Monitor and Alert on Suspicious Activity
  - Log all authentication attempts and monitor for anomalies, such as multiple failed logins from unusual locations.
  - Alert administrators in real-time for suspicious activity, such as brute force attempts or unauthorized access.
- Use Strong Password Policies
  - Enforce complex password requirements (minimum length, uppercase, lowercase, numbers, and special characters).
  - Prohibit the use of common or compromised passwords.
- Implement CAPTCHA for Additional Protection
  - Use CAPTCHA to prevent automated brute force attacks.
- Regular Security Audits and Penetration Testing
  - Conduct regular security audits and penetration testing to identify and address authentication vulnerabilities.
- Implement Zero Trust Architecture
  - Adopt a Zero Trust model to ensure continuous verification of user identity and access rights.

2. Insecure Password Reset Vulnerability in OWASP Account

- **Description:** Insecure password reset mechanisms can be exploited by attackers to gain unauthorized access to user accounts. This vulnerability arises from weak or predictable reset tokens, lack of rate limiting, or insufficient verification processes.
- **Impact:** This vulnerability allows attackers to take over user accounts, leading to unauthorized access to sensitive data and potential account misuse.
- **Severity:** High
- **CVSS Score:** 7.5 (High)

Poc

1. Go to the login page

2. Click Forgot password

Recommendations

- Secure Reset Links Cryptographically With Time-Limited Tokens
  - Generate reset links using cryptographically secure random tokens.
  - Ensure tokens are time-limited (e.g., expire after 15 minutes) and invalidated after use.
- Rotate and Revoke Cookies and Session Keys Securely
  - Rotate session cookies and keys regularly to prevent long-term exploitation if an key is leaked.
  - Revoke all active sessions and cookies when a password reset is requested or when suspicious activity is detected.
- Use HTTPS for Secure Cookie Transmission
  - Enforce HTTPS to prevent cookie interception during transmission.
  - Set the Secure flag for cookies to ensure they are only transmitted over HTTPS.
- Implement Rate Limiting and Account Lockout Mechanisms
  - Limit the number of password reset requests per IP address and per account to prevent brute force attacks.
  - Temporarily lock accounts after a specified number of failed attempts (e.g., 5 attempts within 15 minutes).
- Log and Monitor Password Reset Activity
  - Log all password reset requests and monitor for anomalies, such as multiple requests from unusual locations or IP addresses.
  - Alert administrators in real-time for suspicious activity, such as brute force attempts or unauthorized access.
- Implement Multi-Factor Authentication (MFA)
  - Enforce MFA for password reset requests to add an extra layer of security.
  - Use trusted authentication apps (Google Authenticator) or hardware tokens for enhanced security.
- Secure Session Management
  - Ensure session tokens are randomly generated and transmitted securely over HTTPS.
  - Implement session timeouts and invalidate sessions after logout or inactivity.
- Use Strong Password Policies
  - Enforce complex password requirements (minimum length, uppercase, lowercase, numbers, and special characters).
  - Prohibit the use of common or compromised passwords.
- Implement CAPTCHA for Additional Protection
  - Use CAPTCHA to prevent automated brute force attacks.
- Regular Security Audits and Penetration Testing
  - Conduct regular security audits and penetration testing to identify and address password reset vulnerabilities.
- Implement Zero Trust Architecture
  - Adopt a Zero Trust model to ensure continuous verification of user identity and access rights.

3. GDPR Data Erasure

- **Description:** Lack of proper data erasure mechanisms can result in non-compliance with data privacy regulations, such as GDPR, leading to potential legal and reputational consequences.
- **Impact:** This vulnerability poses risks to user data privacy and violates their rights to erasure, potentially leading to legal penalties and loss of trust.
- **Severity:** Medium
- **CVSS Score:** 5.3 (Medium)

PoC

1. Go to the profile section of the application

2. Click on Delete account button

3. Intercept the request and capture the response

4. Intruder payload GDPR-REQUEST-PARAMETER (varies by app) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Attack type: Sniper

5. Intruder payload GDPR-REQUEST-PARAMETER = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Attack type: Sniper

6. Intruder payload GDPR-REQUEST-PARAMETER = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Attack type: Sniper

7. Intruder payload GDPR-REQUEST-PARAMETER = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Attack type: Sniper

Recommendations

- Implement Secure Data Erasure Mechanisms
  - Ensure that all user data is permanently erased from all systems and databases upon a verifiable request from the user.
  - Use secure deletion methods that prevent data recovery, such as cryptographic erasure or overwriting data multiple times.
- Implement Rate Limiting and Authentication for Data Erasure Requests
  - Limit the number of data erasure requests per user to prevent abuse.
  - Require strong authentication (e.g., multi-factor authentication) for data erasure requests.
- Log and Monitor Data Erasure Activities
  - Maintain logs of all data erasure requests and actions for auditing and compliance purposes.
  - Monitor logs for anomalies, such as multiple erasure requests from unusual locations or IP addresses.
- Ensure Compliance with GDPR Requirements
  - Provide users with clear information about their rights to erasure and the process for exercising these rights.
  - Ensure that erasure requests are processed within the timeframe specified by GDPR (e.g., 30 days).
- Implement Data Minimization Practices
  - Collect only the minimum data necessary for the application to function, reducing the scope of data subject to erasure requests.
  - Regularly review and purge outdated or unnecessary data to minimize the risk of non-compliance.
- Implement Secure Backup and Recovery Processes
  - Ensure that backups are encrypted and access-controlled to prevent unauthorized access.
  - Implement a secure process for restoring data from backups while ensuring compliance with erasure requests.
- Conduct Regular Security Audits and Penetration Testing
  - Perform regular security audits and penetration testing to identify and address vulnerabilities in the data erasure process.
  - Use third-party auditors to ensure independent verification of compliance and security measures.
- Implement Zero Trust Architecture
  - Adopt a Zero Trust model to ensure continuous verification of user identity and access rights.
  - Limit access to data erasure functionalities based on the principle of least privilege.

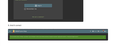
4. Insecure Password Reset Vulnerability in Jim's Account

- **Description:** Insecure password reset mechanisms can be exploited by attackers to gain unauthorized access to user accounts. This vulnerability arises from weak or predictable reset tokens, lack of rate limiting, or insufficient verification processes.
- **Impact:** This vulnerability allows attackers to take over user accounts, leading to unauthorized access to sensitive data and potential account misuse.
- **Severity:** High
- **CVSS Score:** 7.5 (High)

PoC

1. Go to profile of Jim(Victim)

2. Click on Change password button



3. Enter a new password and click on Save button

4. Intercept the request and capture the response

Recommendations

- Secure Reset Links Cryptographically
  - Generate reset links using cryptographically secure random tokens.
  - Ensure tokens are time-limited (e.g., expire after 15 minutes) and invalidated after use.
- Rotate and Revoke Cookies and Session Keys Securely
  - Rotate session cookies and keys regularly to prevent long-term exploitation if an key is leaked.
  - Revoke all active sessions and cookies when a password reset is requested or when suspicious activity is detected.
- Use HTTPS for Secure Cookie Transmission
  - Enforce HTTPS to prevent cookie interception during transmission.
  - Set the Secure flag for cookies to ensure they are only transmitted over HTTPS.
- Implement Rate Limiting and Account Lockout Mechanisms
  - Limit the number of password reset requests per IP address and per account to prevent brute force attacks.
  - Temporarily lock accounts after a specified number of failed attempts (e.g., 5 attempts within 15 minutes).
- Log and Monitor Password Reset Activity
  - Log all password reset requests and monitor for anomalies, such as multiple requests from unusual locations or IP addresses.
  - Alert administrators in real-time for suspicious activity, such as brute force attempts or unauthorized access.
- Implement Multi-Factor Authentication (MFA)
  - Enforce MFA for password reset requests to add an extra layer of security.
  - Use trusted authentication apps (Google Authenticator) or hardware tokens for enhanced security.
- Secure Session Management
  - Ensure session tokens are randomly generated and transmitted securely over HTTPS.
  - Implement session timeouts and invalidate sessions after logout or inactivity.
- Use Strong Password Policies
  - Enforce complex password requirements (minimum length, uppercase, lowercase, numbers, and special characters).
  - Prohibit the use of common or compromised passwords.
- Implement CAPTCHA for Additional Protection
  - Use CAPTCHA to prevent automated brute force attacks.
- Regular Security Audits and Penetration Testing
  - Conduct regular security audits and penetration testing to identify and address password reset vulnerabilities.
- Implement Zero Trust Architecture
  - Adopt a Zero Trust model to ensure continuous verification of user identity and access rights.

5. Login Bypass

- **Description:** Session tokens are used to maintain user sessions after authentication. If session tokens are not properly invalidated upon logout, they can be reused by attackers to gain unauthorized access to the system.
- **Impact:** This vulnerability allows attackers to bypass authentication mechanisms and access sensitive data or perform actions as a legitimate user.
- **Severity:** High
- **CVSS Score:** 7.5 (High)

PoC

1. Go to the login page and enter valid credentials

2. After login click "Sign Out", you get an error such as Bad Request Error page

3. Now in Burp suite go to Intercept and turn on the intercept button

4. Click on go button

5. Now in Burp suite go to Intercept and turn off the intercept button

6. Click on Sign out button

7. Now you are successfully logged out

8. Click on back button and you will see you are still logged in

Recommendations

- Secure Session Management
  - Implement secure session management practices, including:
    - Generating random and unpredictable session tokens.
    - Ensuring session tokens are transmitted over HTTPS.
    - Setting appropriate cookie attributes (Secure, HttpOnly, SameSite).
  - Invalidate sessions on logout and after a specified inactivity period.
- Implement Rate Limiting and Account Lockout Mechanisms
  - Limit the number of login attempts per IP address and per account to prevent brute force attacks.
  - Temporarily lock accounts after a specified number of failed attempts (e.g., 5 attempts within 15 minutes).
- Log and Monitor Login Activity
  - Log all authentication attempts and monitor for anomalies, such as multiple failed logins from unusual locations or IP addresses.
  - Alert administrators in real-time for suspicious activity, such as brute force attempts or unauthorized access.
- Use Strong Password Policies
  - Enforce complex password requirements (minimum length, uppercase, lowercase, numbers, and special characters).
  - Prohibit the use of common or compromised passwords.
- Implement CAPTCHA for Additional Protection
  - Use CAPTCHA to prevent automated brute force attacks.
- Regular Security Audits and Penetration Testing
  - Conduct regular security audits and penetration testing to identify and address authentication vulnerabilities.
- Implement Zero Trust Architecture
  - Adopt a Zero Trust model to ensure continuous verification of user identity and access rights.

6. Change Bender's Password

- **Description:** Session tokens are used to maintain user sessions after authentication. If session tokens are not properly invalidated upon password change, they can be reused by attackers to gain unauthorized access to the system.
- **Impact:** This vulnerability allows attackers to bypass authentication mechanisms and access sensitive data or perform actions as a legitimate user.
- **Severity:** High
- **CVSS Score:** 7.5 (High)

PoC

1. Go to profile of Bender(Victim) - 1 - 1 - 1

Improper Input Validation

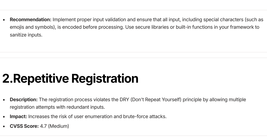
1. Missing Encoding

- **Description:** URL parameters, including emojis and special characters (e.g., #), were not encoded correctly.
- **Impact:** This can lead to improper processing of requests, affecting both security and application behavior.
- **CVSS Score:** 5.3 (Medium).

PoC:

‖ [cat emoji] becomes %F0%9f%b8%bC.

‖ # becomes %23

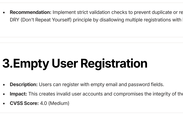


- **Recommendation:** Implement proper input validation and ensure that all input, including special characters (such as emojis and symbols), is encoded before processing. Use secure libraries or built-in functions in your framework to sanitize inputs.

2.Repetitive Registration

- **Description:** The registration process violates the DRY (Don't Repeat Yourself) principle by allowing multiple registration attempts with redundant inputs.
- **Impact:** Increases the risk of user enumeration and brute-force attacks.
- **CVSS Score:** 4.7 (Medium)

PoC:

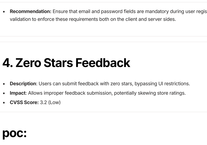


- **Recommendation:** Implement strict validation checks to prevent duplicate or redundant registrations. Enforce the DRY (Don't Repeat Yourself) principle by disallowing multiple registrations with identical input.

3.Empty User Registration

- **Description:** Users can register with empty email and password fields.
- **Impact:** This creates invalid user accounts and compromises the integrity of the registration process.
- **CVSS Score:** 4.0 (Medium)

poc:

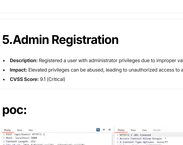


- **Recommendation:** Ensure that email and password fields are mandatory during user registration. Implement input validation to enforce these requirements both on the client and server sides.

4. Zero Stars Feedback

- **Description:** Users can submit feedback with zero stars, bypassing UI restrictions.
- **Impact:** Allows improper feedback submission, potentially skewing store ratings.
- **CVSS Score:** 3.2 (Low)

poc:

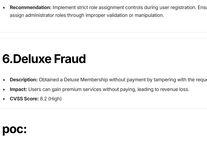


- **Recommendation:** Ensure server-side validation to prevent feedback with zero stars. Enforce minimum input values on both client and server sides to ensure data integrity.

5.Admin Registration

- **Description:** Registered a user with administrator privileges due to improper validation.
- **Impact:** Elevated privileges can be abused, leading to unauthorized access to administrative functions.
- **CVSS Score:** 9.1 (Critical)

poc:

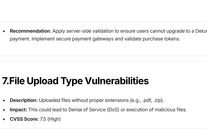


- **Recommendation:** Implement strict role assignment controls during user registration. Ensure that users cannot self-assign administrator roles through improper validation or manipulation.

6.Deluxe Fraud

- **Description:** Obtained a Deluxe Membership without payment by tampering with the request parameters.
- **Impact:** Users can gain premium services without paying, leading to revenue loss.
- **CVSS Score:** 8.2 (High)

poc:

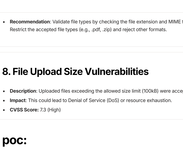


- **Recommendation:** Apply server-side validation to ensure users cannot upgrade to a Deluxe Membership without payment. Implement secure payment gateways and validate purchase tokens.

7.File Upload Type Vulnerabilities

- **Description:** Uploaded files without proper extensions (e.g., .pdf, .zip).
- **Impact:** This could lead to Denial of Service (DoS) or execution of malicious files.
- **CVSS Score:** 7.5 (High)

poc:

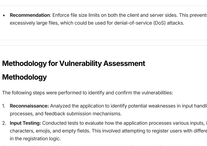


- **Recommendation:** Validate file types by checking the file extension and MIME type during the upload process. Restrict the accepted file types (e.g., .pdf, .zip) and reject other formats.

8. File Upload Size Vulnerabilities

- **Description:** Uploaded files exceeding the allowed size limit (100kB) were accepted.
- **Impact:** This could lead to Denial of Service (DoS) or resource exhaustion.
- **CVSS Score:** 7.3 (High)

poc:



- **Recommendation:** Enforce file size limits on both the client and server sides. This prevents users from uploading excessively large files, which could be used for denial-of-service (DoS) attacks.

Methodology for Vulnerability Assessment

Methodology

The following steps were performed to identify and confirm the vulnerabilities:

1. **Reconnaissance:** Analyzed the application to identify potential weaknesses in input handling, user registration processes, and feedback submission mechanisms.
2. **Input Testing:** Conducted tests to evaluate how the application processes various inputs, including special characters, emojis, and empty fields. This involved attempting to register users with different inputs to identify flaws in the registration logic.
3. **Payload Injection:** Utilized various payloads, such as URLs with emojis and special characters, to verify the encoding processes within the application. This included testing the system's behavior with invalid or malicious inputs.
4. **Validation Checks:** Examined the application for proper validation mechanisms to prevent repetitive registration, empty registrations, and zero-star feedback submissions. Attempted to register users with duplicate information and incomplete forms.
5. **Privilege Escalation Testing:** Tested the registration process for users to see if administrator roles could be assigned improperly, including attempts to manipulate request parameters to gain elevated privileges.
6. **Transaction Manipulation:** Attempted to bypass payment processes by altering request parameters to gain access to premium features or memberships without proper authorization.
7. **File Upload Testing:** Evaluated file upload functionality by attempting to upload files with unsupported extensions or exceeding size limits. This included testing for both client-side and server-side validation of file types and sizes.
8. **Documentation:** Recorded findings, including vulnerability descriptions, potential exploit methods, and impacts on the application's security posture.
9. **Reporting:** Compiled a detailed report summarizing the identified vulnerabilities, their implications, and specific recommendations for mitigation.

Made with Gamma

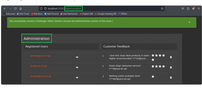
Broken Access Control

1. Admin Section Access Vulnerability

- **Description**: Unauthorized access to the Admin Section was identified.
- **Impact**: Allows an attacker to access sensitive admin features, compromising the application's integrity.
- **CVSS Score**: 8.0 (High)

PoC:

1. Login with the admin
2. Guessing that the admin's section in the path /administration



Mitigation Recommendations:

- **Implement Role-Based Access Control (RBAC)**: Ensure that only authorized users with specific roles (such as "Admin") have access to the admin section.
- **Strict Authentication**: Enforce strict authentication mechanisms like multi-factor authentication (MFA) for administrative accounts.
- **Session Validation**: Implement session tokens that are validated against user roles before granting access to admin features.
- **Least Privilege Principle**: Assign the minimum set of permissions necessary for each role, ensuring non-admin users have no access to the admin section.
- **Audit Logs**: Enable detailed logging for access to the admin section to monitor and detect any unauthorized access attempts.

Methodology

- **Common Path Enumeration**: Attackers often start by attempting to access common admin paths like /admin, /administration, /administrator, /dashboard, or /control panel by manually entering these URLs in the browser.
- **Error Messages Analysis**: By observing error messages when accessing restricted paths, an attacker can gain hints about the existence of admin sections. For example, a "403 Forbidden" error may indicate a valid path.
- **Directory Brute-Forcing**: Tools like Dirbuster, Dirsearch, or Burp Suite can be used to brute-force directories and files in the application to find hidden or unlinked admin pages.
- **Inspection of HTML/JavaScript**: Attackers can inspect the source code or JavaScript files for clues pointing to admin paths.
- **Social Engineering**: Attackers may use phishing or other social engineering tactics to trick a legitimate admin into revealing the correct admin URL.

2. View Basket Vulnerability

- **Description**: Unauthorized access to the View Basket functionality was discovered.
- **Impact**: Allows attackers to alter basket items and quantities, potentially leading to financial loss.
- **CVSS Score**: 6.8 (Medium)

PoC:

1. Click on "Your Basket"

2. Go to burp http history and select this request then send to repeater and found that the request contain the User'id" "1" change it to "2" another User'id and send the request

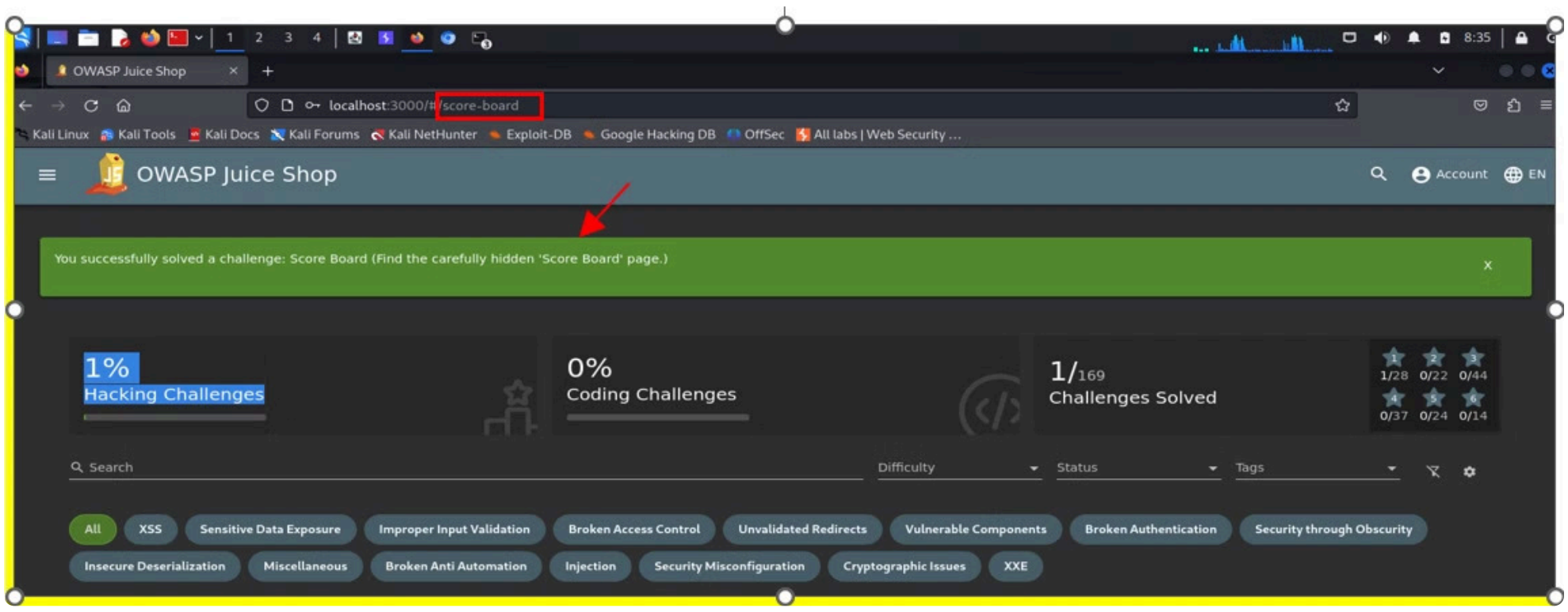
Request
Response
GET /rest/basket/1 HTTP/1.1
Host: localhost:3000
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIm5leHRQYXNzd29yZCI6ImRlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsIm5leHRQYXNzd29yZCI6IiIsIm5leHRQYXNzd29yZCI6IiIsIm5leHRQYXNzd29yZ

Miscellaneous

1. Score Board Access Vulnerability

- **Description**: Direct access to the Score Board feature was identified without authentication.
- **Impact**: Reveals insights into security challenges, aiding further exploitation.
- **CVSS Score**: 5.5 (Medium)

PoC:



- **Recommendation**:

1. **Authentication Requirement**: Implement authentication checks for accessing the Score Board feature to ensure that only authorized users can view it.
2. **Access Control**: Utilize role-based access control (RBAC) to restrict access based on user roles and permissions.
3. **Audit Logs**: Maintain detailed audit logs of access to the Score Board to monitor for unauthorized access attempts.
4. **Security Awareness**: Regularly review and update access policies to ensure compliance with security best practices.

2. Miscellaneous Privacy Policy Vulnerability

- **Description**: The Privacy Policy page can be accessed without proper security measures.
- **Impact**: Unauthorized access to privacy-related information could aid in social engineering attacks.
- **CVSS Score**: 3.7 (Low)
- **Recommendation**:

1. **Regular Review**: Conduct regular reviews of the privacy policy to ensure it aligns with current legal requirements and best practices (e.g., GDPR, CCPA).
2. **Transparency**: Clearly inform users about data collection, usage, and sharing practices in the privacy policy.
3. **User Consent**: Ensure that user consent is obtained for data collection and processing, and provide options for users to manage their privacy settings.
4. **Data Protection Measures**: Implement robust data protection measures, including encryption and access controls, to safeguard user data.
5. **Training and Awareness**: Train employees on privacy policies and data protection practices to foster a culture of compliance and awareness within the organization.

Made with Gamma