

Web Application Security Assessment Report – OWASP Juice Shop

*Created By :- Nishant
Date :- 08 Sept 2025
Internship Program: Future
Interns Cyber Security
Internship*

Web Application Security Assessment Report – OWASP Juice Shop

Web Application Security Testing. The purpose of this assessment is to evaluate the security posture of the **OWASP Juice Shop**, an intentionally vulnerable application widely used for training and practice in penetration testing.

The testing methodology followed the **OWASP Top 10 (2021) security risks** and included the use of industry-standard tools such as **OWASP ZAP** for automated scanning, **Burp Suite Community Edition** for manual testing, and **Nikto** for server misconfiguration analysis. During the course of this project, multiple vulnerabilities were identified, including **SQL Injection, Broken Access Control, Reflected XSS, and Information Disclosure** through exposed directories.

The findings presented in this report include **detailed descriptions of each vulnerability, reproduction steps, screenshots, associated OWASP Top 10 mappings, impact analysis, and recommended mitigations**. This deliverable simulates a real-world client security assessment and is intended to demonstrate practical skills in ethical hacking, vulnerability assessment, and penetration testing.

Scope and Methodology

The scope of this assessment was limited to the **OWASP Juice Shop application**, a deliberately insecure web application designed for security training. The objective was to perform a **vulnerability assessment and penetration test** based on OWASP Top 10 security risks.

Tools Used

- **OWASP ZAP** – Automated vulnerability scanning
- **Burp Suite Community Edition** – Manual exploitation of vulnerabilities
- **Nikto** – Web server misconfiguration scanning
- **Browser Developer Tools / Manual Payloads** – For custom testing (SQLi, XSS)

Testing Methodology

The methodology adopted for this assessment was based on industry standards and the OWASP Testing Guide:

1. **Reconnaissance & Information Gathering** – Identified technologies and potential attack surfaces.
2. **Automated Scanning** – Used OWASP ZAP and Nikto to detect possible vulnerabilities.
3. **Manual Testing** – Performed SQL injection, XSS, and access control testing using Burp Suite.
4. **Vulnerability Analysis** – Validated findings and mapped them to OWASP Top 10.
5. **Reporting** – Documented vulnerabilities with screenshots, impacts, and mitigations.

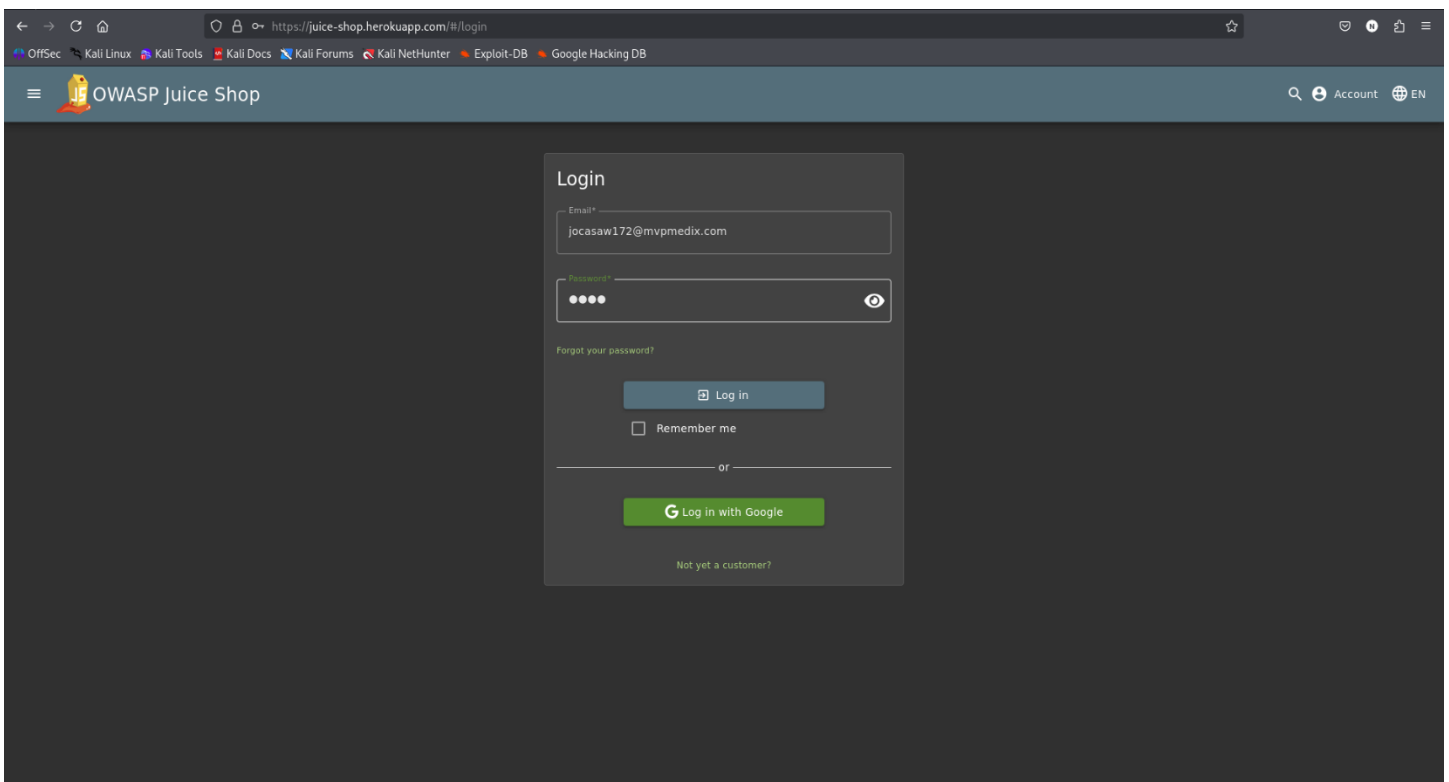
Finding 1: SQL Injection – Authentication Bypass

Severity: High

Description: During testing of the login functionality, it was identified that the application is vulnerable to SQL Injection. By inserting a specially crafted payload in the username field, the authentication query was manipulated, allowing login without valid credentials.

Steps with Evidence

Step 1: Opened the login page and entered random credentials in the email and password fields. The login attempt failed as expected.



Step 2: Intercepted the login request using **Burp Suite**. The request contained the email and password parameters.

Intercepted request details for `https://juice-shop.herokuapp.com/rest/user/login`.

Time	Type	Direction	Method	URL	Status code	Length
01:25:31 3 Sept 2...	HTTP	→ Request	POST	https://juice-shop.herokuapp.com/rest/user/login		
01:25:31 3 Sept 2...	HTTP	→ Request	GET	https://juice-shop.herokuapp.com/rest/user/whoami		
01:25:31 3 Sept 2...	HTTP	→ Request	GET	https://juice-shop.herokuapp.com/rest/user/whoami		

Request

```
1 POST /rest/user/login HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=PyuOtY1KSkTgcJ1rT2a3fphPhDtDxulktm71lEf0suvRheatkpcL71VkrZ6Fz1tLQco0UqwbtkacPESzaF4EfKNH2JTE4H51
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/json
9 Content-Length: 53
10 Origin: https://juice-shop.herokuapp.com
11 Referer: https://juice-shop.herokuapp.com/
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Priority: u=0
16 Te: trailers
17 Connection: keep-alive
18 {
19   "email": "jocasaw172@mvpmidix.com",
20   "password": "1234"
21 }
```

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 4
- Request headers: 16

Step 3: Modified the email parameter to include a SQL Injection payload:

' OR 1=1--

This payload forces the query condition to always return true.

Intercepted request details for `https://juice-shop.herokuapp.com/rest/user/login`.

Time	Type	Direction	Method	URL	Status code	Length
01:32:24 3 Sept 2...	HTTP	→ Request	GET	https://juice-shop.herokuapp.com/socket.io/?EIO=4&transport=polling&sid=P40KVgGr6JkkWpADCP		
01:32:24 3 Sept 2...	WS	→ To server		https://juice-shop.herokuapp.com/socket.io/?EIO=4&transport=websocket&sid=P40KVgGr6JkkWpADCP		6
01:32:42 3 Sept 2...	HTTP	→ Request	GET	https://juice-shop.herokuapp.com/rest/user/whoami		
01:32:42 3 Sept 2...	HTTP	→ Request	GET	https://juice-shop.herokuapp.com/rest/user/whoami		
01:32:42 3 Sept 2...	HTTP	→ Request	POST	https://juice-shop.herokuapp.com/rest/user/login		
01:32:52 3 Sept 2...	HTTP	→ Request	POST	https://juice-shop.herokuapp.com/socket.io/?EIO=4&transport=polling&sid=P40KVgGr6JkkWpADCP		
01:32:54 3 Sept 2...	HTTP	→ Request	GET	https://juice-shop.herokuapp.com/socket.io/?EIO=4&transport=polling&sid=P40KVgGr6JkkWpADCP		
01:32:59 3 Sept 2...	HTTP	→ Request	POST	https://chatgpt.com/backend-api/conversation/implicit_message_feedback		

Request

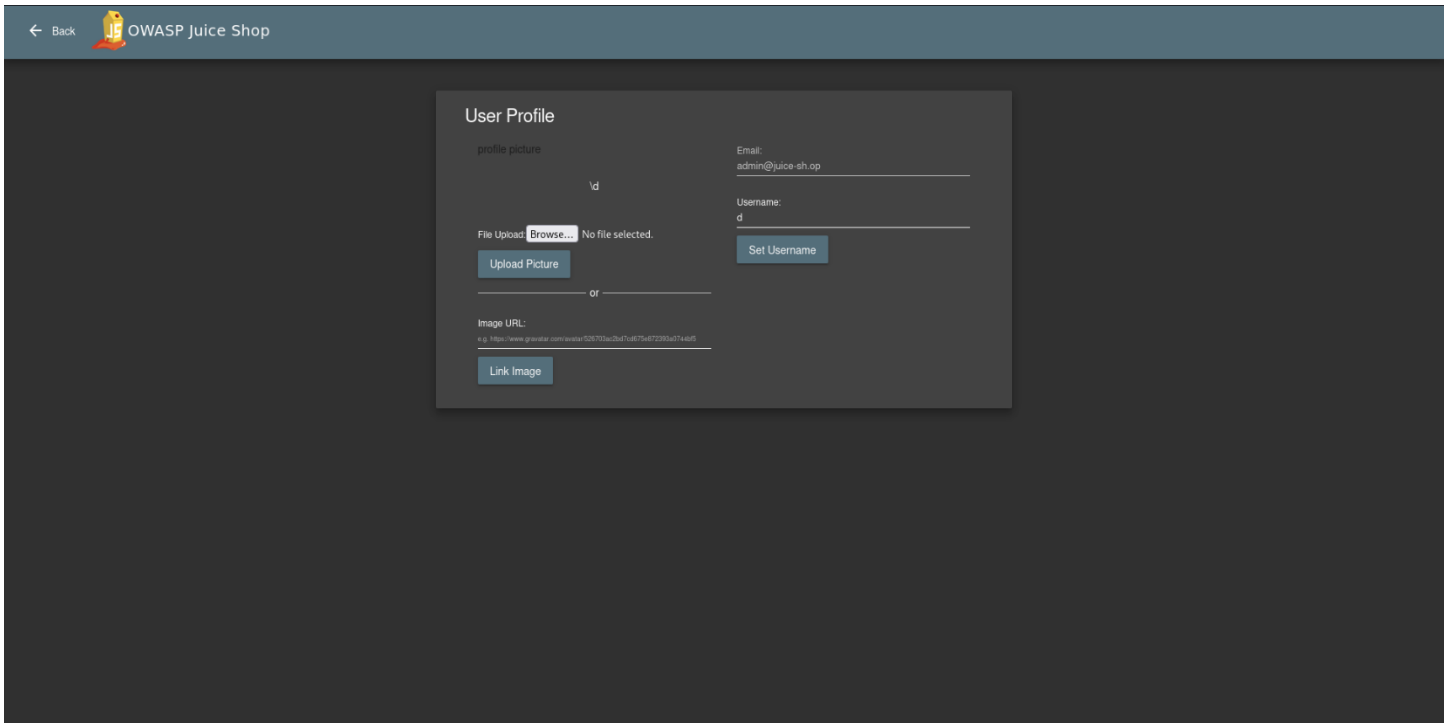
```
1 POST /rest/user/login HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=PyuOtY1KSkTgcJ1rT2a3fphPhDtDxulktm71lEf0suvRheatkpcL71VkrZ6Fz1tLQco0UqwbtkacPESzaF4EfKNH2JTE4H51
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/json
9 Content-Length: 53
10 Origin: https://juice-shop.herokuapp.com
11 Referer: https://juice-shop.herokuapp.com/
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Priority: u=0
16 Te: trailers
17 Connection: keep-alive
18 {
19   "email": "' OR 1=1--",
20   "password": "1234"
21 }
```

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 4
- Request headers: 16

Step 4: Forwarded the modified request to the server. Instead of rejecting, the application processed the login and redirected to the authenticated area.

Step 5: Verified that the login was successful and administrative access was granted without valid credentials.



Impact:

- Full administrative access can be achieved.
- Attackers may extract sensitive information from the database.
- Leads to complete compromise of the application and stored user data.

Mitigation:

- Use parameterized queries (Prepared Statements).
- Implement strong server-side input validation and sanitization.
- Apply the principle of least privilege on database accounts.

OWASP Top 10 Mapping:

- A03: Injection

Finding 2: Broken Access Control – Saved Cards Misuse

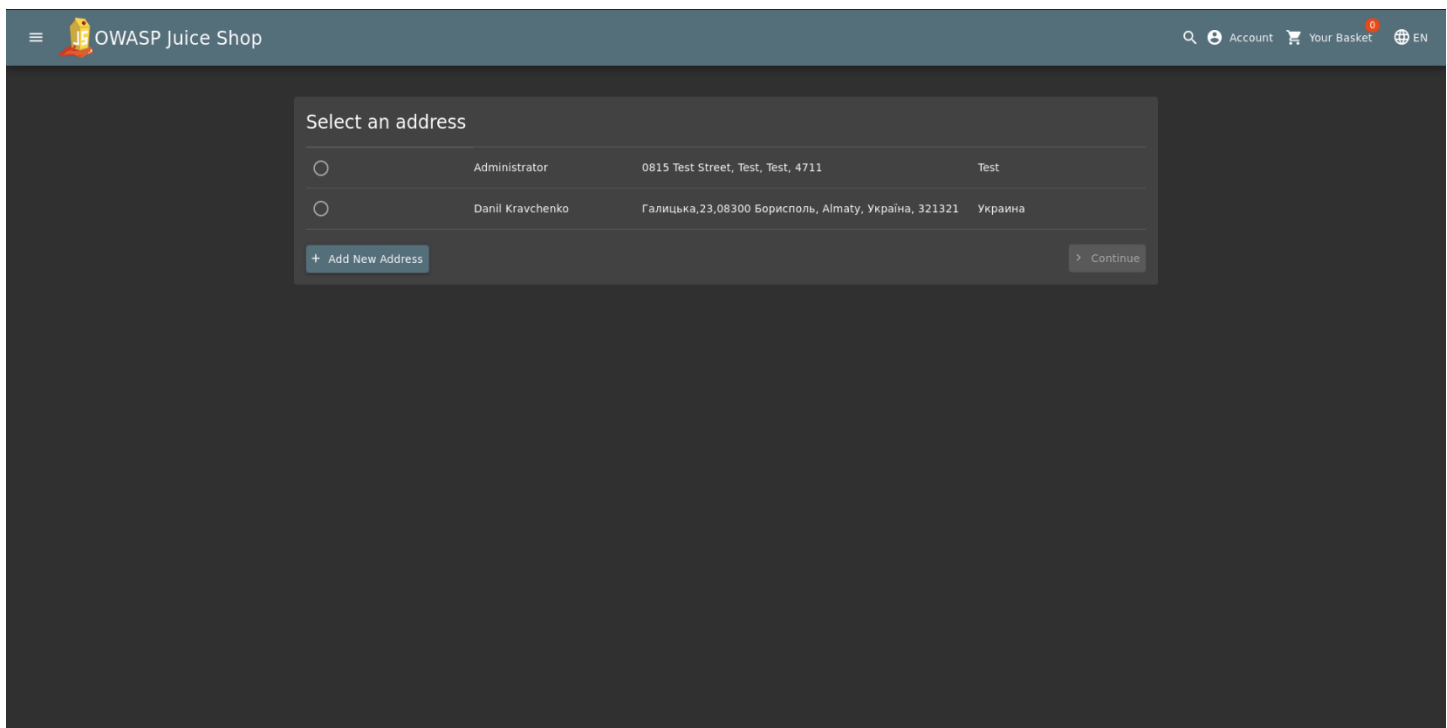
Severity: High

Description:

During testing of the checkout process, it was observed that previously saved payment methods (credit cards and digital wallet) could be used to make purchases **without requiring OTP or CVV re-verification**. This indicates a broken access control and weak payment authorization mechanism.

Steps with Evidence

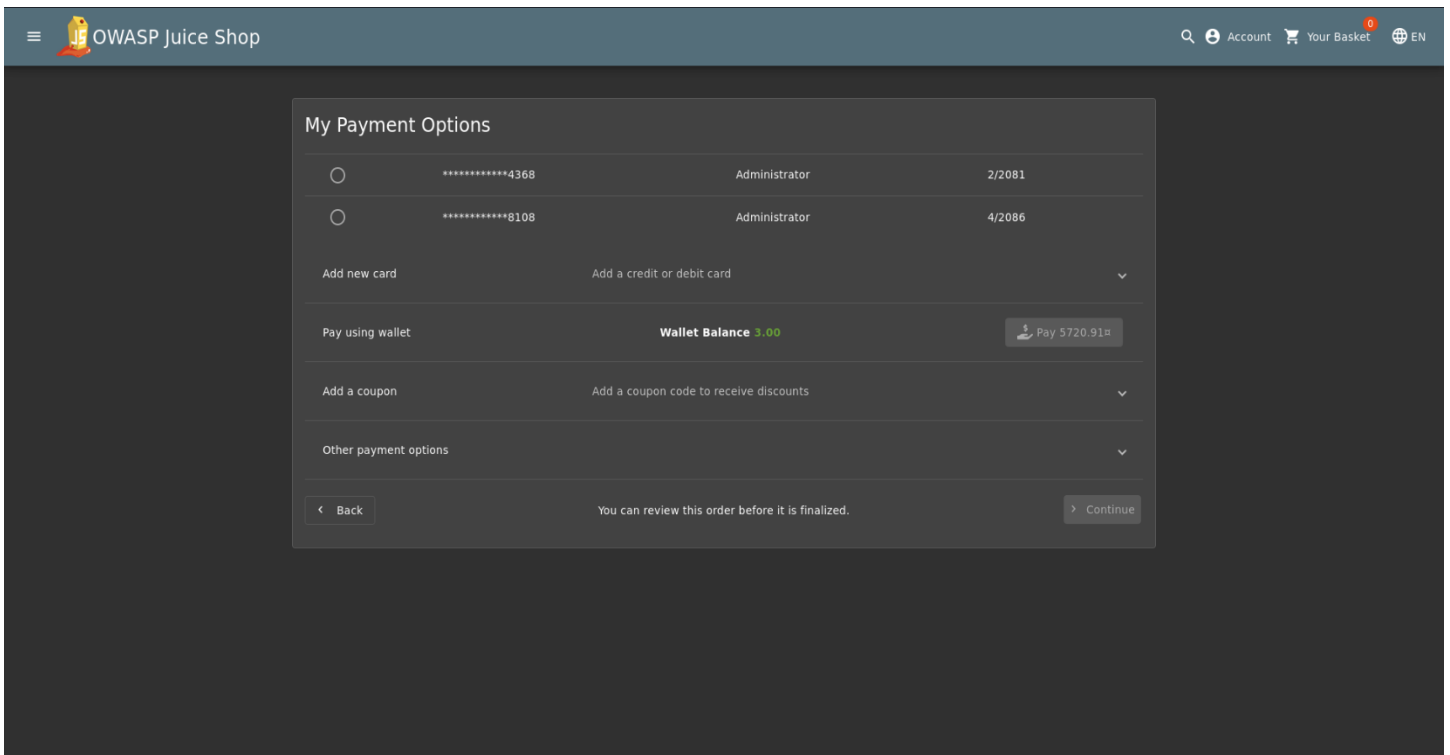
Step 1: Added an item to the shopping cart and proceeded to checkout.



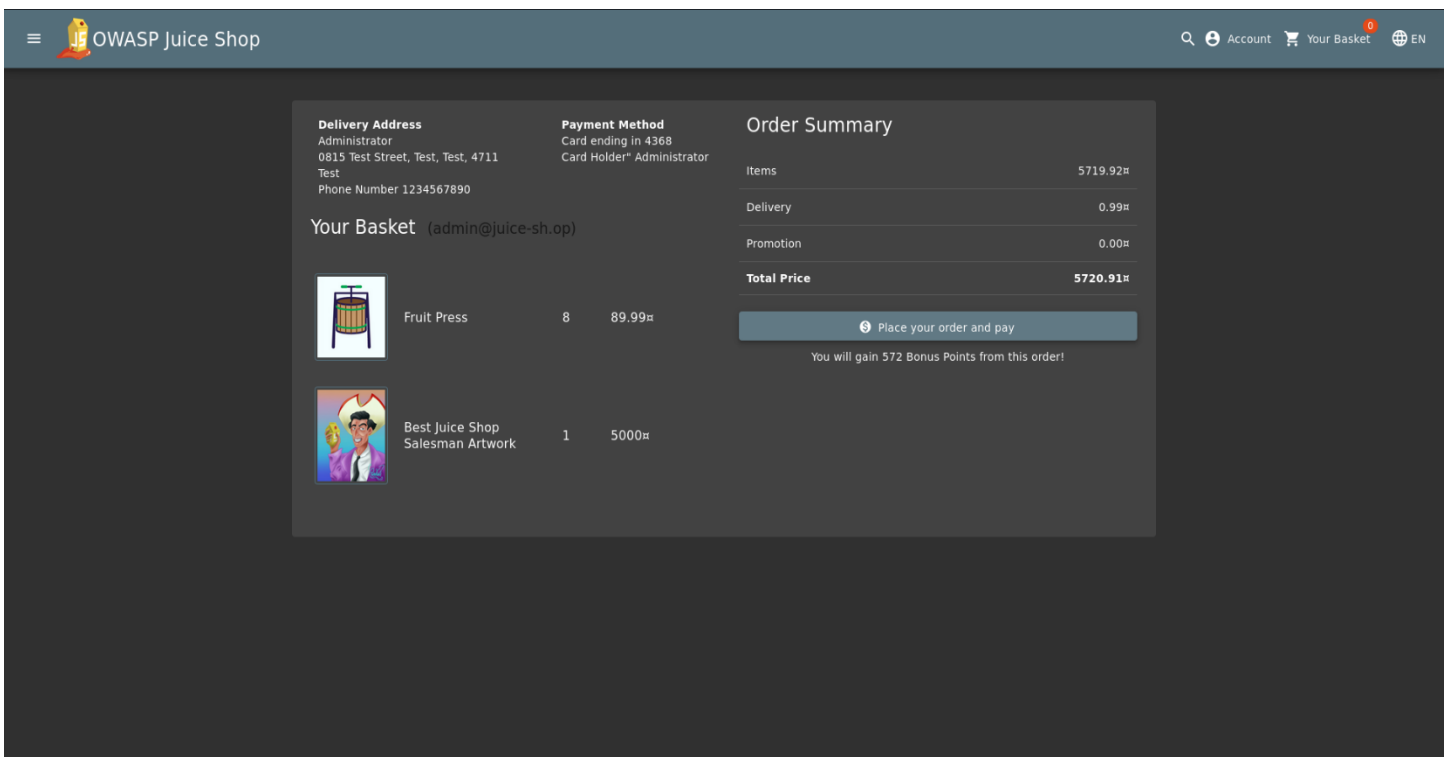
The screenshot shows the OWASP Juice Shop interface. At the top, there is a navigation bar with the shop name, a search icon, and links for Account, Your Basket (with a 0 item count), and a language selector (EN). Below the navigation bar, a modal titled "Select an address" is displayed. This modal contains a table with two rows of saved addresses, each with a radio button for selection. The first row is for "Administrator" with address "0815 Test Street, Test, Test, 4711" and country "Test". The second row is for "Danil Kravchenko" with address "Галицька, 23, 08300 Борисполь, Almaty, Україна, 321321" and country "Україна". At the bottom of the modal, there is a button to "+ Add New Address" and a "Continue" button with a right arrow.

Select an address			
<input type="radio"/>	Administrator	0815 Test Street, Test, Test, 4711	Test
<input type="radio"/>	Danil Kravchenko	Галицька, 23, 08300 Борисполь, Almaty, Україна, 321321	Україна
+ Add New Address		Continue	


Step 2: Reached the payment page. Two saved credit cards and a digital wallet option were available for selection.







Step 3: Selected one of the saved cards and continued. No CVV or OTP verification was required to complete the purchase.



Step 4: Completed the transaction successfully. The application displayed a confirmation message along with the order summary.

 OWASP Juice Shop

  Account  Your Basket  EN



Thank you for your purchase!

Your order has been placed and is being processed. You can check for status updates on our [Track Orders](#) page.

Your order will be delivered in 1 days.

Delivery Address
Administrator
0815 Test Street, Test, Test, 4711
Test
Phone Number 1234567890


Order Summary





 

Product	Price	Quantity	Total Price
Fruit Press	89.99€	8	719.92€
Best Juice Shop Salesman Artwork	5000€	1	5000.00€
		Items	5719.92€
		Delivery	0.99€
		Promotion	0.00€
		Total Price	5720.91€

You have gained 572 Bonus Points from this order!

Step 5: Navigated to the order history page. All past purchases, including the unauthorized transaction, were listed.

 OWASP Juice Shop

  Account  Your Basket  EN



Order History

Order ID
#5267-91583eeb554b681c

Total Price
5720.91€

Bonus
572

In Transit



Product	Price	Quantity	Total Price
Fruit Press	89.99€	8	719.92€
Best Juice Shop Salesman Artwork	5000€	1	5000.00€

Order ID
#5267-8df7eaf8b31dcfd7

Total Price
52.37€

Bonus
3

In Transit



Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99€	5	9.95€
Orange Juice (1000ml)	2.99€	5	14.95€
Eggfruit Juice (500ml)	8.99€	3	26.97€

Order ID
#5267-ed1a7cba0e6bcdb4

Total Price
26.97€

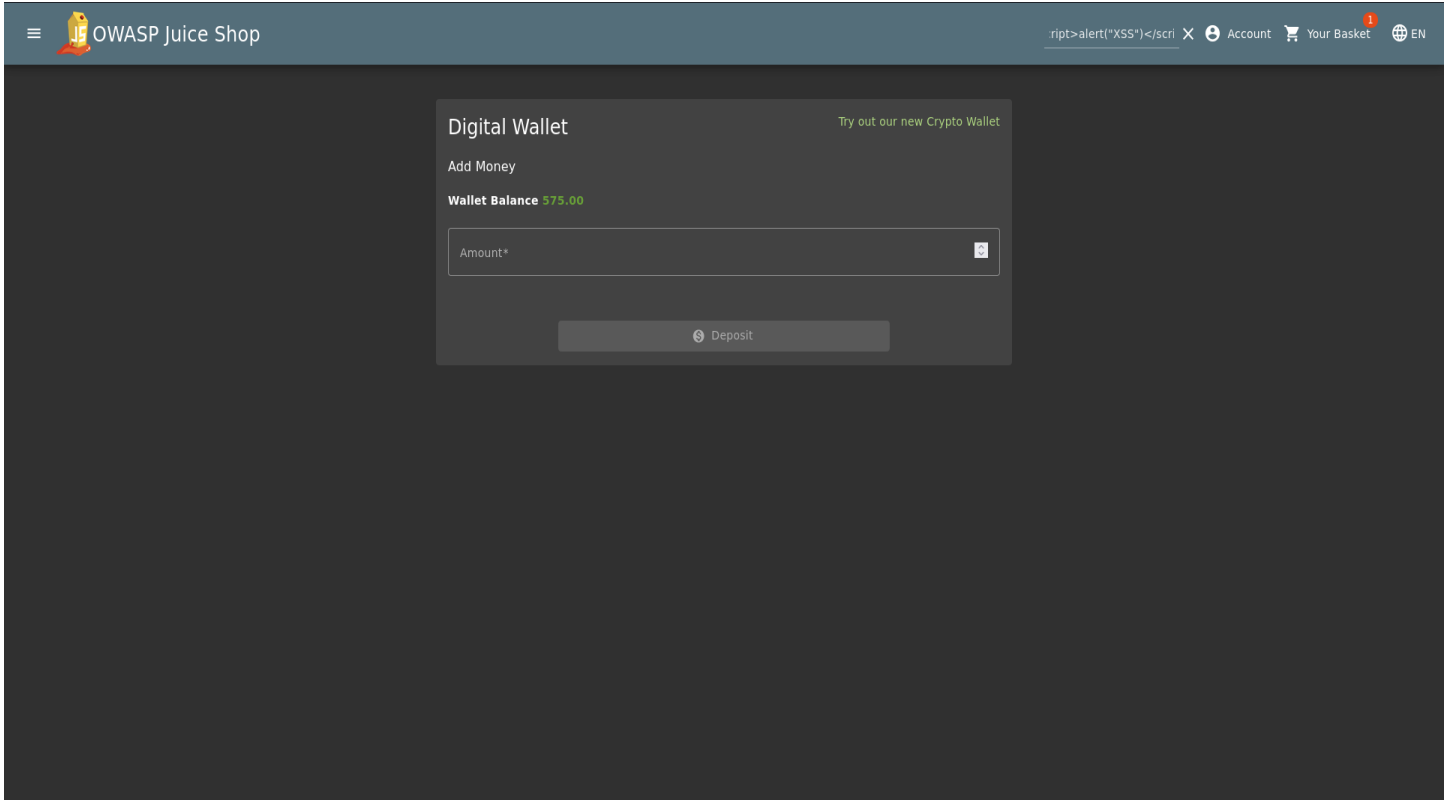
Bonus
3

Delivered

Product	Price	Quantity	Total Price
Eggfruit Juice (500ml)	8.99€	3	26.97€

Step 6: Accessed the digital wallet feature and confirmed that funds could be viewed and added without additional authentication.



Impact:

- Attackers can misuse stored cards for unauthorized purchases.
- Risk of **financial fraud** and **loss of customer trust**.
- May lead to **PCI-DSS non-compliance**.

Mitigation:

- Enforce **re-authentication** (OTP, CVV) for every transaction.
- Mask and encrypt saved card details.
- Implement **role-based access control (RBAC)** to restrict admin misuse.
- Ensure compliance with **PCI-DSS standards**.

OWASP Top 10 Mapping:

- A01: Broken Access Control

Finding 3: Information Disclosure – /ftp Directory Exposure

Severity: Medium

Description: Using **DirBuster**, a hidden /ftp directory was discovered. It exposes sensitive files including backup configs, error logs, encrypted announcements, compiled Python code, and a KeePass database (incident-support.kdbx).

Steps with Evidence

Step 1: Executed a directory brute-force scan with Dirb using a common wordlist.

dirb https://juice-shop.herokuapp.com -o dirb_result.txt

```
(nix@kali)-[~]
$ dirb https://juice-shop.herokuapp.com -o dirb_result.txt

-----
DIRB v2.22
By The Dark Raver
-----

OUTPUT_FILE: dirb_result.txt
START_TIME: Tue Sep  9 13:44:57 2025
URL_BASE: https://juice-shop.herokuapp.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

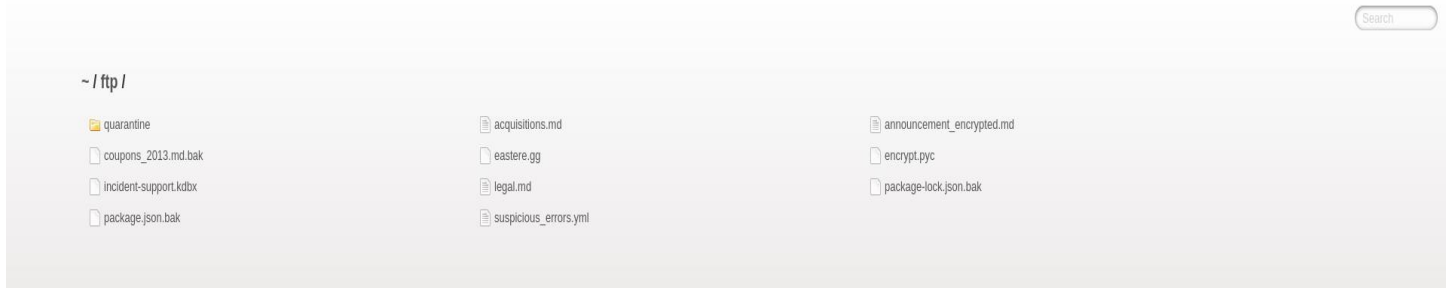
GENERATED WORDS: 4612

---- Scanning URL: https://juice-shop.herokuapp.com/ ----
+ https://juice-shop.herokuapp.com/assets (CODE:301|SIZE:156)
+ https://juice-shop.herokuapp.com/ftp (CODE:200|SIZE:12742)
+ https://juice-shop.herokuapp.com/profile (CODE:500|SIZE:1036)
+ https://juice-shop.herokuapp.com/promotion (CODE:200|SIZE:6586)
+ https://juice-shop.herokuapp.com/redirect (CODE:500|SIZE:2959)
+ https://juice-shop.herokuapp.com/robots.txt (CODE:200|SIZE:28)
+ https://juice-shop.herokuapp.com/video (CODE:200|SIZE:10075518)
+ https://juice-shop.herokuapp.com/Video (CODE:200|SIZE:10075518)

-----

END_TIME: Tue Sep  9 14:09:30 2025
DOWNLOADED: 4612 - FOUND: 8
```

Step 2: Accessed the /ftp directory in the browser. The server displayed an open directory listing with sensitive files.



Step 3: Observed files such as:

- incident-support.kdbx (KeePass DB, may store credentials)
- package.json.bak and package-lock.json.bak (backup configs)
- suspicious_errors.yml (error logs)
- encrypt.pyc (compiled Python code)

Impact:

- Disclosure of configs and credentials.
- Files like .kdbx and .pyc may lead to full compromise.

Mitigation:

- Disable **directory listing** on the web server.
- Remove backup and test files before deploying to production.
- Store sensitive files outside the web root.
- Enforce strict file access permissions.

OWASP Top 10 Mapping:

- A01: Broken Access Control
- A05: Security Misconfiguration

Finding 4: Reflected XSS – Search Bar

Severity: Medium

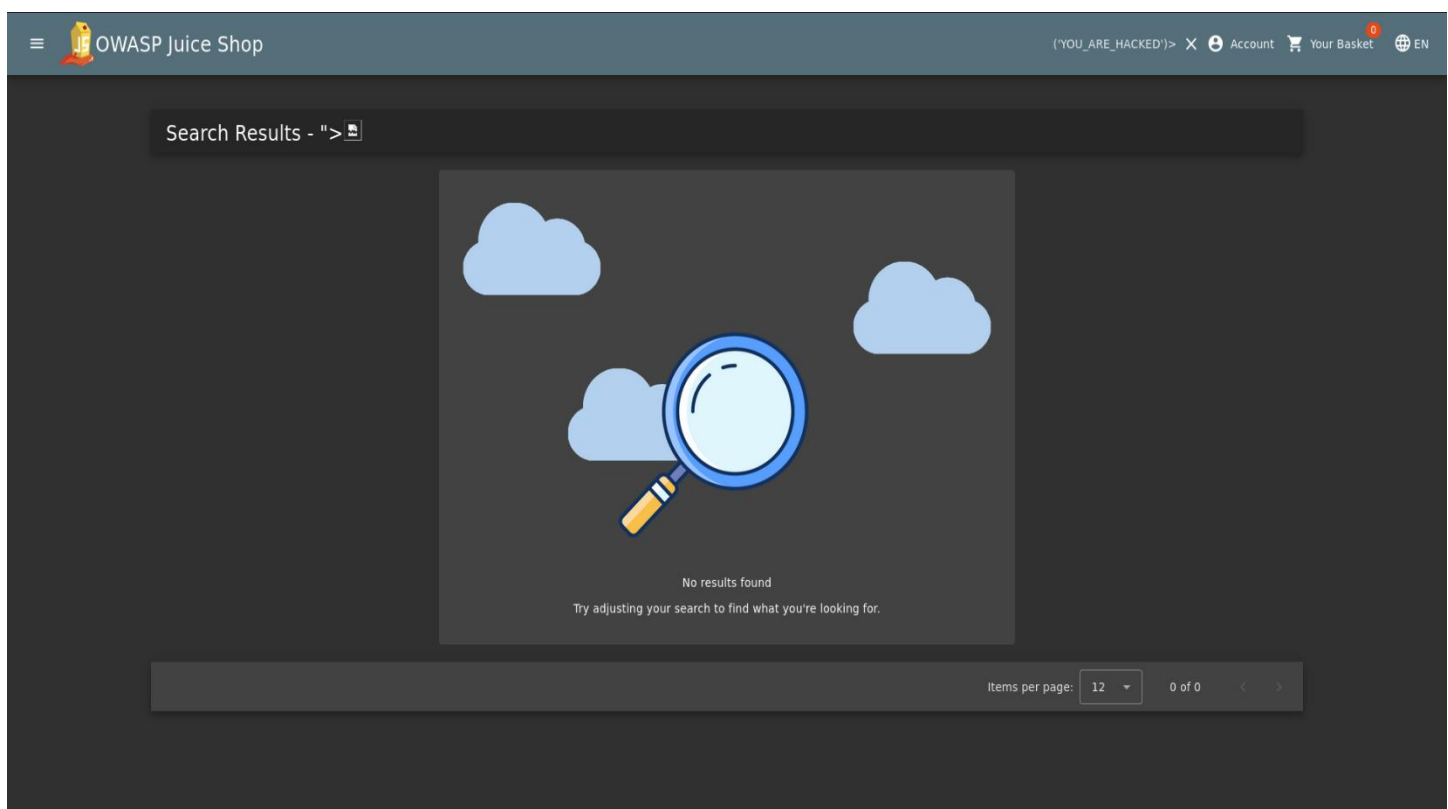
Description:

The search functionality reflects unsensitized user input in the response. By injecting a crafted payload, JavaScript code was executed in the browser, confirming a Reflected Cross-Site Scripting (XSS) vulnerability. This type of issue can be exploited by attackers through malicious links, making unsuspecting users run harmful scripts in their browsers.

Steps with Evidence

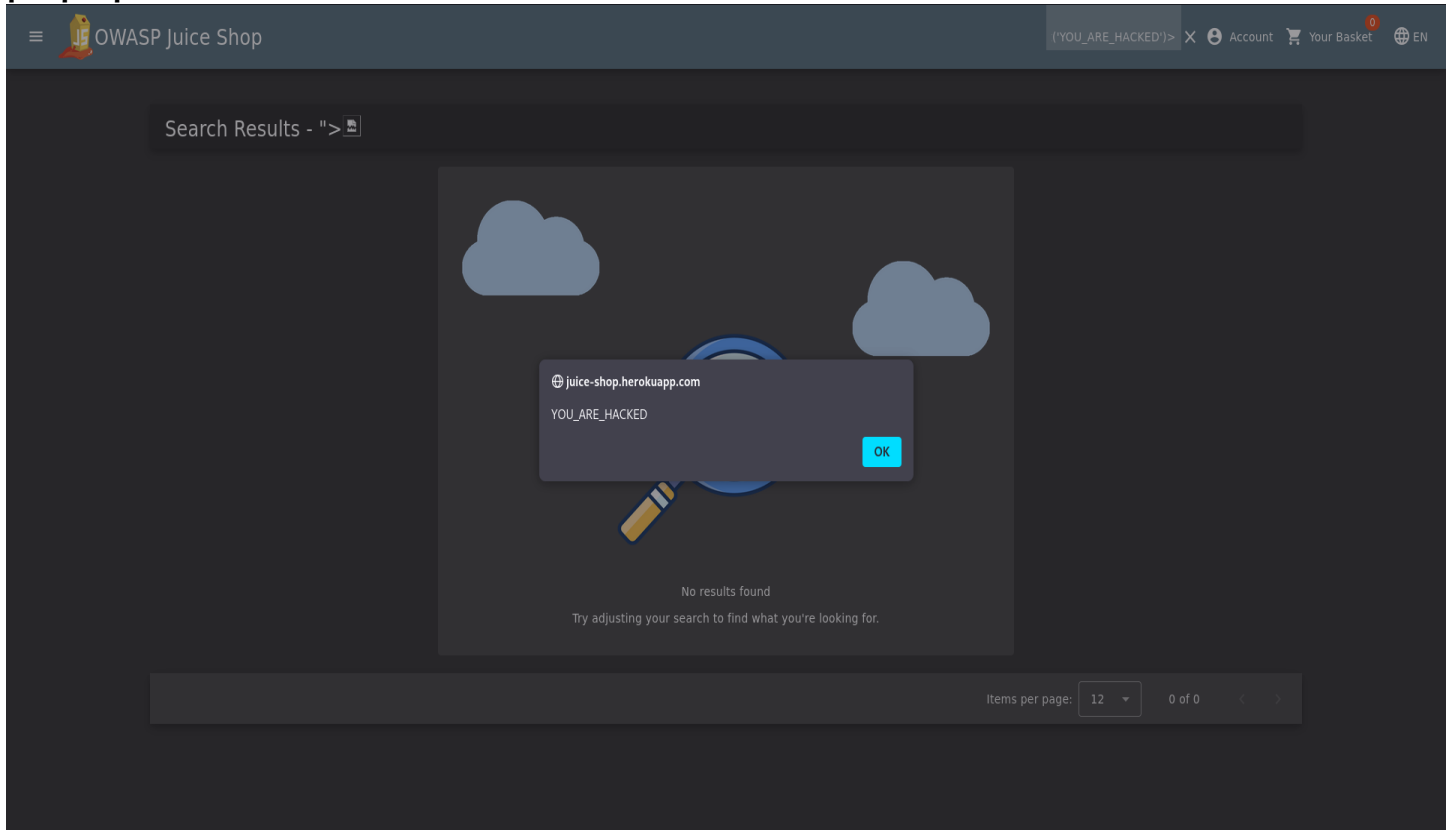
Step 1: Navigated to the application's search bar and entered a crafted XSS payload:

```
"><img src=x onerror=alert('YOU_ARE_HACKED')>
```



Step 2: Submitted the search request. The payload was reflected back in the response without sanitization.

Step 3: The browser executed the injected JavaScript, displaying a popup alert box.



Impact:

- Attackers can steal session cookies.
- Users may be redirected to phishing sites.
- Enables account compromise if exploited with social engineering.

Mitigation:

- Sanitize and encode user inputs before rendering.
- Implement a strong Content Security Policy (CSP).
- Use libraries like DOMPurify for input validation.

OWASP Mapping:

- A03: Injection

Finding 5: Automated Scan – OWASP ZAP

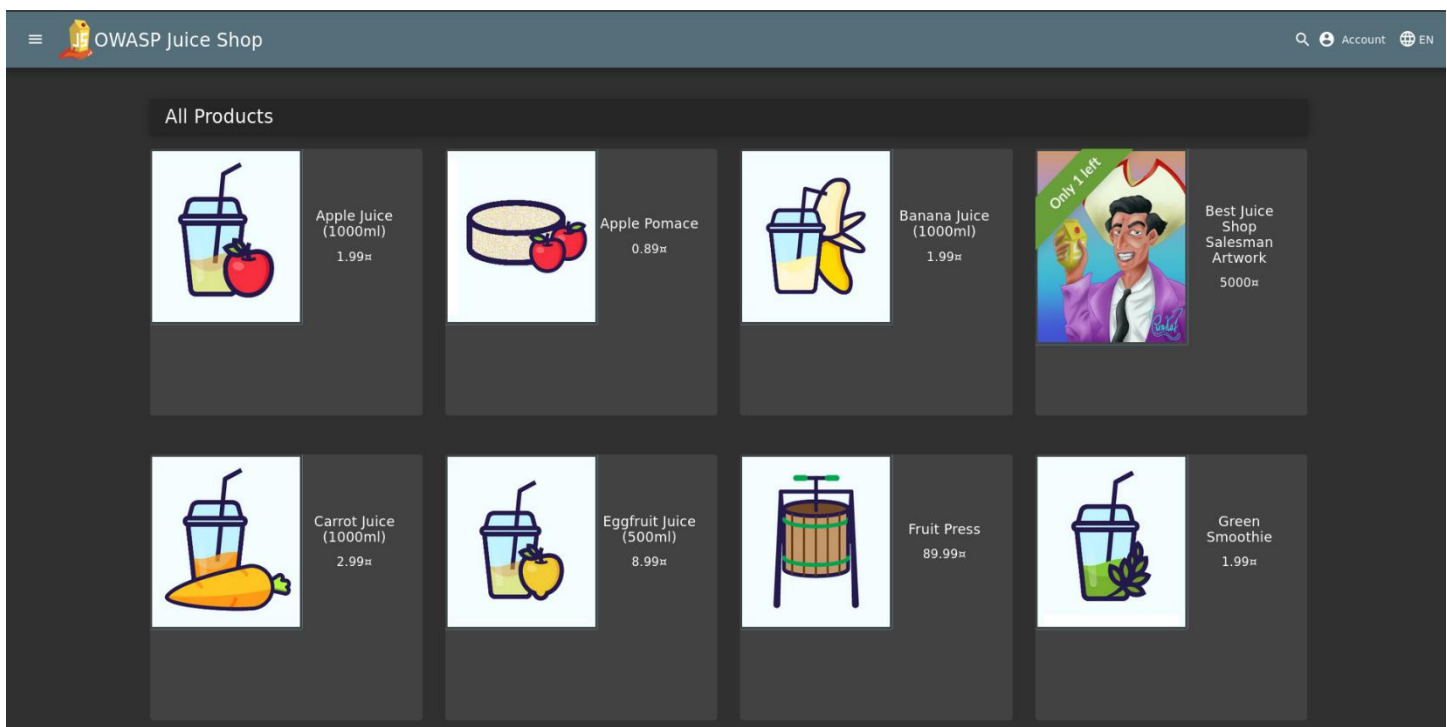
Severity: Informational

Description:

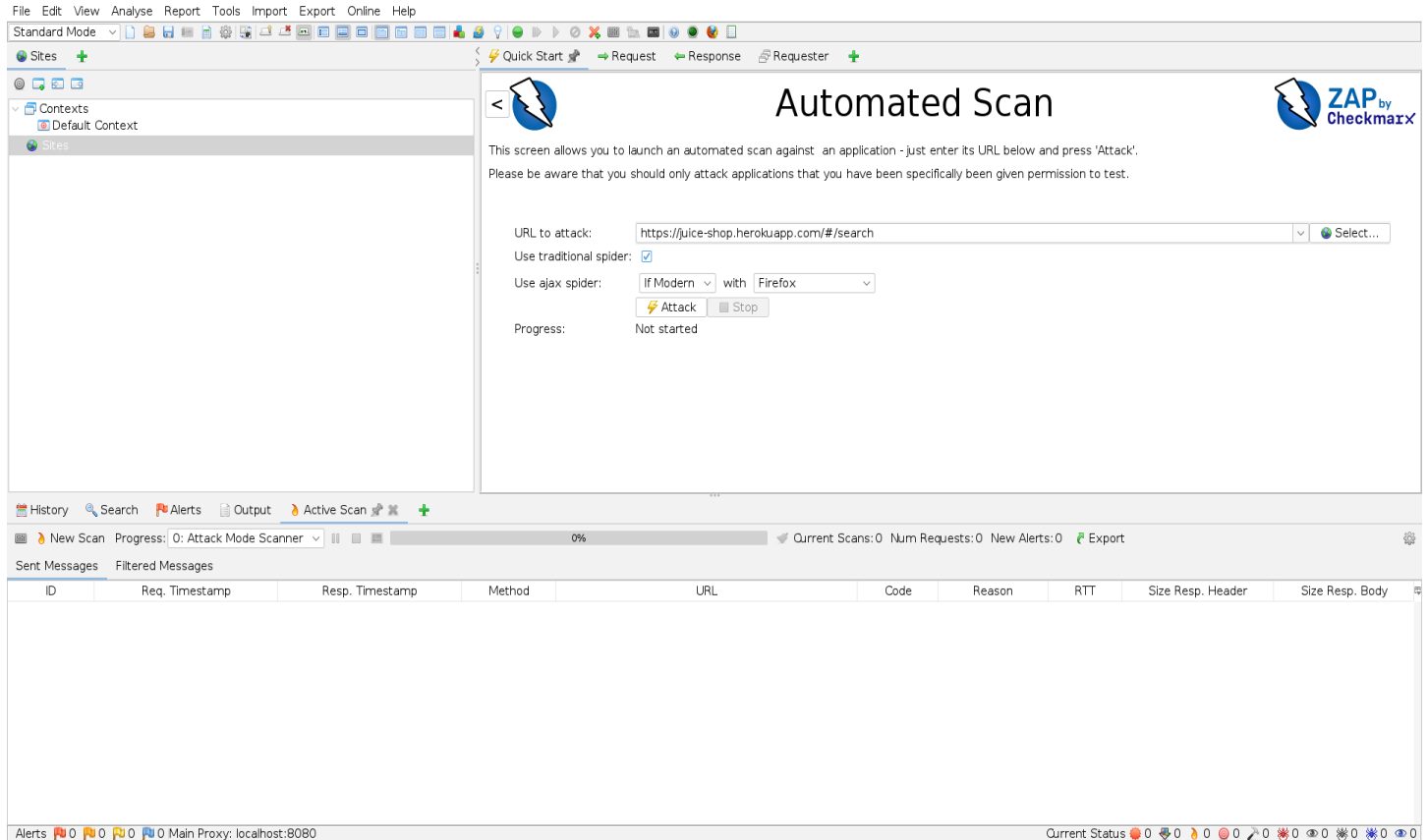
An automated vulnerability scan was performed using OWASP ZAP against the OWASP Juice Shop application. The scan identified several potential issues, including possible SQL Injection, XSS, and security misconfigurations. These results were further analyzed and manually validated with Burp Suite to confirm actual exploitable vulnerabilities.

Steps with Evidence

Step 1: Launched OWASP ZAP and provided the target Juice Shop application URL for scanning. The target application's initial view was verified.



Step 2: Entered the Juice Shop URL into the **Automated Scan** section of ZAP's Quick Start tab.



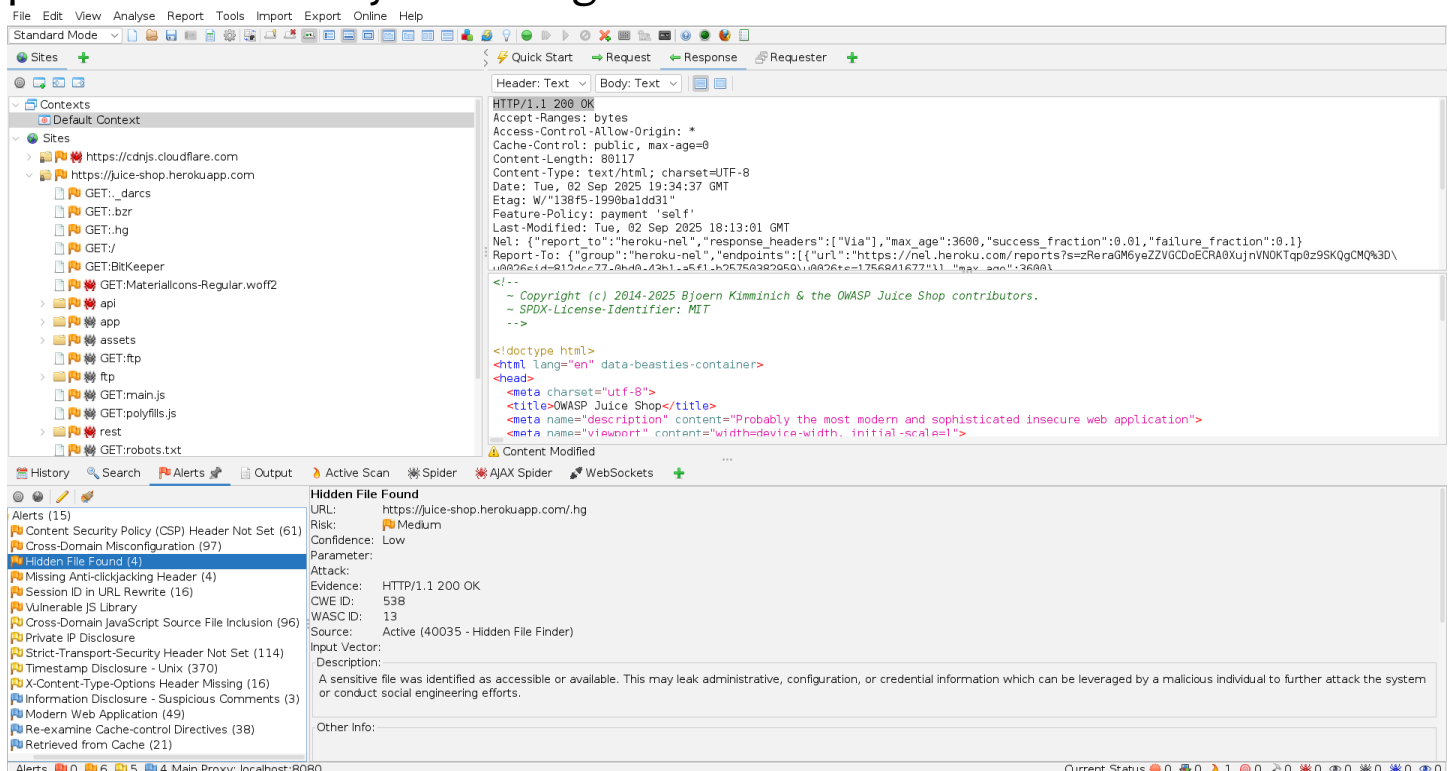
The screenshot shows the ZAP (Zed Attack Proxy) interface in the 'Quick Start' tab, specifically the 'Automated Scan' section. The interface is titled 'Automated Scan' and includes a ZAP logo. Below the title, there is a brief instruction: 'This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'. Please be aware that you should only attack applications that you have been specifically given permission to test.'

The main configuration area includes the following fields and controls:

- URL to attack:** A text input field containing 'https://juice-shop.herokuapp.com/#/search'.
- Use traditional spider:** A checkbox that is checked.
- Use ajax spider:** A dropdown menu set to 'If Modern' with a 'with' dropdown set to 'Firefox'.
- Attack/Stop buttons:** Two buttons, 'Attack' (highlighted with a lightning bolt icon) and 'Stop'.
- Progress:** A label indicating 'Not started'.

At the bottom of the interface, there is a progress bar showing '0%' and a status bar with various icons and text: 'Alerts 0', 'New Scan Progress: 0: Attack Mode Scanner', 'Current Scans: 0', 'Num Requests: 0', 'New Alerts: 0', and 'Export'.

Step 3: After the scan completed, ZAP displayed a list of potential vulnerabilities in the **Alerts** tab. These included possible injection points and security misconfigurations.



The screenshot shows the ZAP interface with the 'Alerts' tab selected. The left sidebar displays a list of alerts, with 'Hidden File Found (4)' highlighted. The main pane shows the details of the selected alert, including the URL, risk level, confidence, and a description of the vulnerability.

Alerts (15)

- Content-Security-Policy (CSP) Header Not Set (61)
- Cross-Domain Misconfiguration (97)
- Hidden File Found (4)**
- Missing Anti-clickjacking Header (4)
- Session ID in URL Rewrite (16)
- Vulnerable JS Library
- Cross-Domain JavaScript Source File Inclusion (96)
- Private IP Disclosure
- Strict-Transport-Security Header Not Set (114)
- Timestamp Disclosure - Unix (370)
- X-Content-Type-Options Header Missing (16)
- Information Disclosure - Suspicious Comments (3)
- Modern Web Application (49)
- Re-examine Cache-control Directives (38)
- Retrieved from Cache (21)

Hidden File Found

URL: https://juice-shop.herokuapp.com/hg

Risk: Medium

Confidence: Low

Parameter:

Attack:

Evidence: HTTP/1.1 200 OK

CWE ID: 538

WASC ID: 13

Source: Active (40035 - Hidden File Finder)

Input Vector:

Description:

A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.

Other Info:

Impact:

- Provides an initial vulnerability overview.
- Helps identify potential weak points quickly.
- Automated results should not be considered final without manual verification.

Mitigation:

- Regularly perform automated scans with OWASP ZAP.
- Always combine automated scanning with manual penetration testing for accuracy.

OWASP Mapping:

- Multiple categories (A01: Broken Access Control, A03: Injection, A05: Security Misconfiguration) depending on flagged issue.

Finding 6: Nikto Scan – Informational Results

Severity: Informational

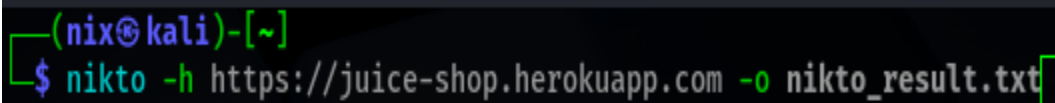
Description:

A Nikto web server scan was performed against the OWASP Juice Shop application to identify server misconfigurations, outdated components, and known vulnerabilities. The scan completed successfully but did not reveal any critical security issues.

Steps with Evidence

Step 1: Executed the Nikto scan against the target URL:

nikto -h <https://juice-shop.herokuapp.com> -o nikto_result.txt

A terminal window with a dark background. The prompt is (nix@kali)-[~]. The command entered is nikto -h https://juice-shop.herokuapp.com -o nikto_result.txt. The cursor is at the end of the command.

```
(nix@kali)-[~]  
$ nikto -h https://juice-shop.herokuapp.com -o nikto_result.txt
```

Step 2: The scan results listed several observations, including:

- Presence of unusual HTTP headers (Cross-Domain, Reporting-Endpoints).
- Missing **X-Content-Type-Options** header.
- Robots.txt file exposing /ftp path.
- Multiple references to potential backup/certificate files.

```

[nix@kali]~$ nikto -h https://juice-shop.herokuapp.com -o nikto_result.txt
- Nikto v2.3.0
-----
+ Multiple IPs found: 46.137.15.86, 54.220.192.176, 54.73.53.134
+ Target IP: 46.137.15.86
+ Target Hostname: juice-shop.herokuapp.com
+ Target Port: 443
-----
+ SSL Info: Subject: /CN=*.herokuapp.com
Ciphers: ECDHE-RSA-AES128-GCM-SHA256
Issuer: /C=US/O=Amazon/CN=Amazon RSA 2048 M02
+ Start Time: 2025-09-09 13:45:55 (GMT+5)
-----
+ Server: Heroku
+ /: Retrieved via header: 1.1 heroku-router.
+ /: Retrieved access-control-allow-origin header: *.
+ /: Uncommon header 'x-recruiting' found, with contents: /#/jobs.
+ /: Uncommon header 'reporting-endpoints' found, with contents: heroku-nel="https://nel.heroku.com/reports?s=KPJb0FMJGYl5xqIpiSB0WJ0tUew925NrvJUGzJwHVEK3D6sId=812dcc77-0bd0-43b1-a5f1-b257503829596ts=1757405759".
+ /: The site uses TLS and the Strict-Transport-Security HTTP header is not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /robots.txt: Entry '/ftp/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: contains 1 entry which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Server is using a wildcard certificate: *.herokuapp.com. See: https://en.wikipedia.org/wiki/Wildcard_certificate
+ /dump.jks: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ : Server banner changed from 'Heroku' to 'heroku-router'.
+ /46.137.15.86.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.jks: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.jks: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /com.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /site.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /46.137.15.86.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /site.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /46.137.15.86.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /com.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /database.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.jks: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /backup.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /herokuapp.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /juice-shop.herokuapp.com.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /com.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html

```

Impact:

- No exploitable high or medium-severity issues were identified.
- General server information such as headers and directories were listed.

Mitigation:

- Regularly update web server and application dependencies.
- Periodically rerun vulnerability scans as part of routine security testing.

OWASP Mapping:

- Not directly mapped – Informational only.

Conclusion

The security assessment of the OWASP Juice Shop application was conducted as part of the Future Interns Cyber Security Internship Program (Task 1). The testing combined automated scans (OWASP ZAP, Nikto) with manual verification (Burp Suite, payload testing).

A total of 4 confirmed vulnerabilities were identified:

- SQL Injection (Authentication Bypass)
- Broken Access Control (Saved Cards Misuse)
- Information Disclosure (/ftp Directory)
- Reflected Cross-Site Scripting (XSS in Search Bar)

These issues highlight weaknesses in input validation, access control, and server configuration. While the application is intentionally vulnerable, the findings demonstrate the importance of secure coding practices and robust configuration management.

The application resisted CSRF attacks, and Nikto scans did not reveal critical findings, showing some protective mechanisms are in place. Implementing the suggested mitigations will significantly reduce the risk of exploitation and strengthen the security posture of the application.

OWASP Top 10 Mapping

Vulnerability	Severity	OWASP Top 10 Mapping
SQL Injection (Login Bypass)	High	A03: Injection
Saved Cards Misuse (Broken Access)	High	A01: Broken Access Control
/ftp Directory Exposure	Medium	A05: Security Misconfiguration
Reflected XSS (Search Bar)	Medium	A03: Injection
Owasp Zap Scan Results	Informational	Not Applicable (No critical findings)
Nikto Scan Results	Informational	Not Applicable (No critical findings)

Final Recommendations & Next Steps

Based on the findings from this security assessment, the following recommendations are provided to improve the security posture of the OWASP Juice Shop application and prevent real-world exploitation:

General Recommendations

- **Input Validation:** Apply strict server-side validation and sanitization for all user inputs to prevent SQL Injection and XSS.
- **Authentication & Access Control:** Enforce re-authentication for sensitive actions and ensure proper role-based access restrictions.
- **Configuration Hardening:** Remove backup/test files and disable directory listing to prevent information disclosure.
- **Encryption Practices:** Securely store sensitive data (such as saved cards) and ensure compliance with data protection standards.
- **Regular Updates & Patching:** Keep all frameworks, libraries, and server components up to date with the latest security patches.

Next Steps

- Conduct a retest after implementing the recommended fixes to validate remediation.
- Integrate security testing tools (ZAP, Burp, Nikto) into the development lifecycle (DevSecOps approach).
- Perform periodic penetration testing to identify new vulnerabilities.
- Provide security awareness training for developers to align with OWASP secure coding practices.



Thank You!