TechMagic

Penetration Test Report

Sample

OWASP juice-shop

February 10th, 2019

▸ ross@techmagic.co TechMagic LLC ▸ https://techmagic.co/

Table of content

Executive summary

We conducted a pentest (vulnerability assessment) of the OWASP juice shop application in order to detect existing web application vulnerabilities and determine exposure to a targeted attack. All activities were conducted in a manner that simulated a real threat actor engaged in a targeted attack against OWASP juice shop application with the next goals:

- Identifying if a remote attacker could penetrate application defenses
- Determining the impact of a security breach on:
    - Confidentiality of the company's private data
    - Application infrastructure and its availability
    - Confidentiality of the users private information
- Identifying infrastructure issues
- Identifying web application vulnerabilities that could lead to:
    - Unauthorized access to confidential data
    - Web application users data leak
    - Web application DB enumeration in order to retrieve confidential data
    - Web application crash
    - Exploitation of vulnerable third-party components used by the application
    - Malware campaign or targeted attack against web application users with consequent spear-phishing or SE (Social Engineering) attacks

Efforts were placed on the identification and exploitation of security weaknesses that could allow a user with limited privileges to access sensitive data, organizational confidential information, backup files, .etc. The attacks were conducted by taking into account the check according to the access matrix of all user roles in order to determine mistakes that could make web applications vulnerable to different kinds of attacks (OWASP Top 10 vulnerabilities list [1]). The assessments were conducted in accordance with the recommendations provided in PTES [2] and OWASP Testing Checklist [3] with all tests and actions being conducted under controlled conditions.

Findings severity ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering or some actions from the end-users. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities in the low range typically have very little impact on an organization's business. The exploitation of such vulnerabilities usually requires local or physical system access.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Scope

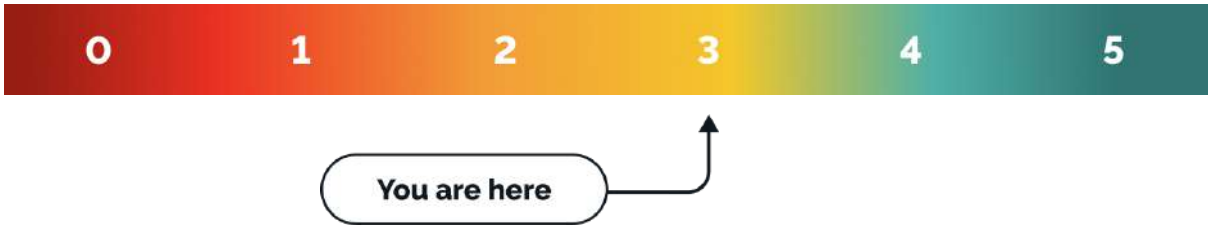
Assessment	Details
Gray Box Web Application Penetration Test	Domain: ● owasp-juice.shop Subdomains: ● demo.owasp-juice.shop ● admin.owasp-juice.shop

Overall security level of the targets in scope

The following domains/subdomains were assessed during the engagement:

- owasp-juice.shop
- demo.owasp-juice.shop

The overall security level of the tested web application as a result of the penetration test is ***'1'***. Revealed issues could allow an attacker to obtain an access to an account of any person, impersonate other users inside the system, trigger errors by providing invalid input, access files containing sensitive information, .etc.



The security levels below are intended to give a high level relative indication of the position of the application or infrastructure tested in terms of its overall security posture and in relation to other applications/infrastructures that were tested by our team. In the below table score '0' means lack of any protection measures and '5' being maximum possible security level:

Grade	Grade description
5	Application/infrastructure has no identified vulnerabilities; its security mechanisms has been tested and we did not manage to penetrate or bypass them
4	Application/infrastructure has no moderate or high-risk, critical vulnerabilities. Identified low/informational severity issues have very little impact on the applications security and its end users. It is recommended to review them and implement required mitigation actions.
3	Application/infrastructure has no critical or high-risk vulnerabilities, but we have identified several moderate severity issues. These issues can be chained together to conduct an attack with significant consequences for the application and its users or they require extra steps such as social engineering for successful exploitation. There should be created a mitigation plan in order to fix the identified issues and minimize a risk of exploitation.
2	Application/infrastructure contains vulnerabilities that may allow compromise in some circumstances. It is advised to form a plan of action and patch as soon as possible.
1	Application/infrastructure has been partly compromised. It is advised to form a plan of action and patch immediately.
0	Application/infrastructure has been completely compromised. It is advised to form a plan of action and patch immediately.

Please note that the definition of compromise in the above indicator is a wide one and is based on our professional experience taking into account other test applications or infrastructures as well as industry best practices and user expectations. From our own experience, level '5' is usually very rarely reached; most tested applications or infrastructure fall under levels '1', '2' or '3'.

Vulnerabilities by impact

Most of the detected vulnerabilities are related to the *demo.owasp-juice.shop* domain. These include various *Broken Access Control* issues, stored and reflected *XSS (Cross Site Scripting)* attacks in different parts of the application, *unrestricted file upload* issues. Several *SQL injection* issues were detected on the *admin.owasp-juice.shop* domain.

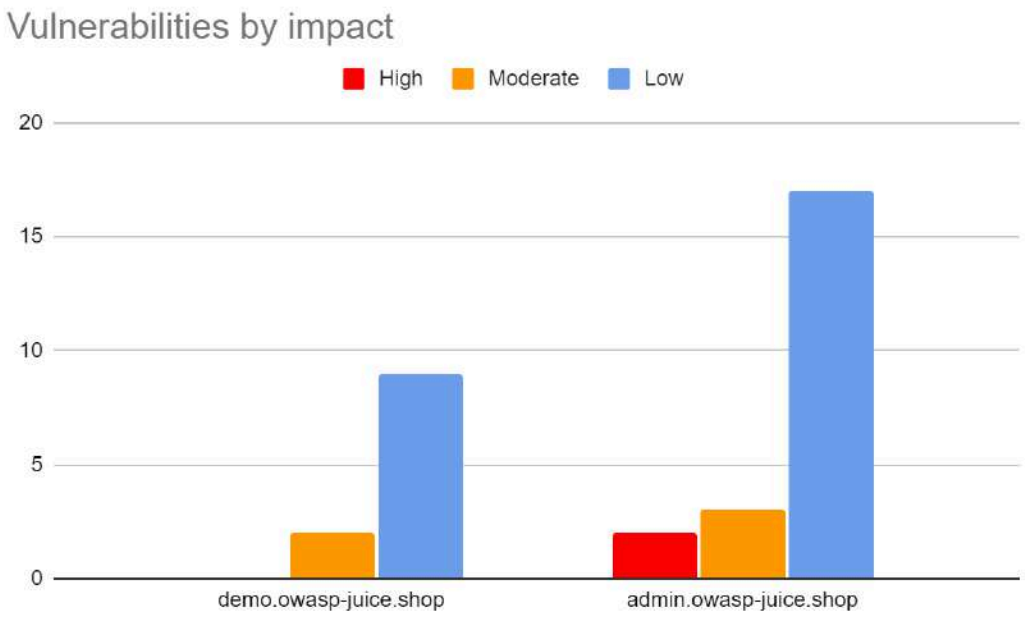


Figure 1.1. Count of detected vulnerabilities by impact.

Successful attacks by type

Most of the detected vulnerabilities are different variants of *Broken Access Control* issues. These vulnerabilities allowed us to retrieve sensitive information about other users. Another dangerous finding is *Unrestricted File Upload* that allows an attacker to upload an arbitrary file to the applications' server.

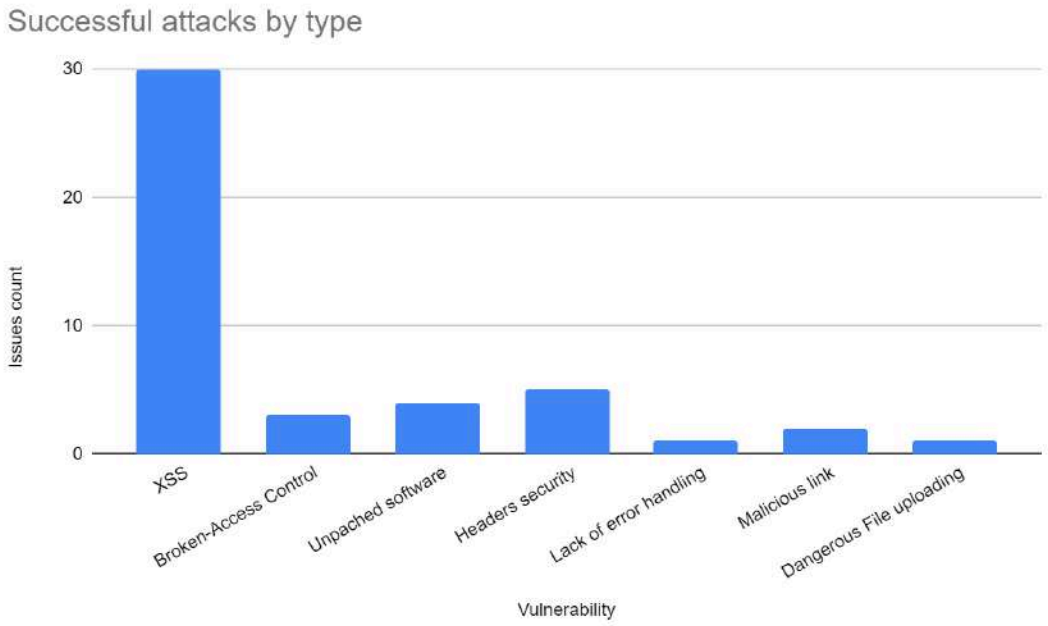


Figure 1.2. Count of successful attacks by type.

Vulnerabilities by cause

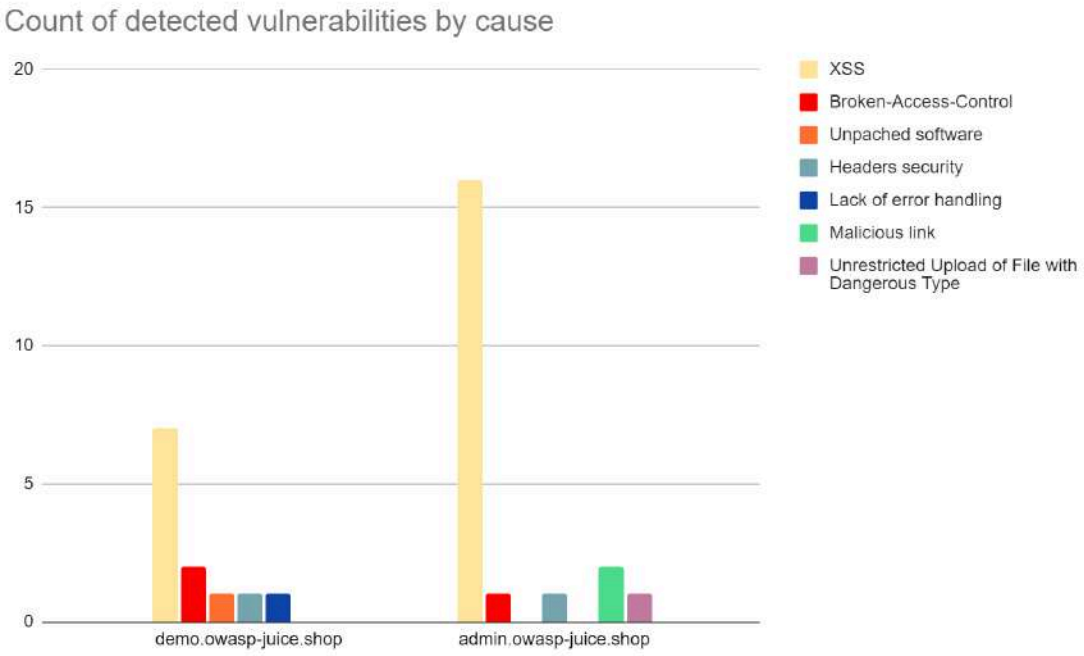


Figure 1.3. Count of successful attacks by type.

Recommendations

Due to the serious impact of disclosed vulnerabilities, appropriate resources should be allocated to mitigate risks and protect the application from malicious threat actors. Future software development processes should be conducted according to SSDLC's (Secure Software Development Lifecycle) best practices to prevent additional attack vectors. The complete list of SSDLC recommendations is listed in the OWASP Secure SDLC cheats sheet series [4].

We recommend the following:

- Fix broken-access control issue according to roles and permissions matrix
- Fix file upload issues, add more checks to prevent uploading of invalid files and links.
- Add client-side validation to prevent XSS vulnerabilities;
- Add strong server-side validation. Because of the high risk of client-side validation bypass, there should be implemented strong server-side validation. It could mitigate such issues as stored XSS, NoSQLi, arbitrary file upload attacks.
- Fix error handling. Disclosed by the application error logs could help an attacker to exploit the vulnerability and fix issues in the submitted payload, enumerate valid usernames, roles, .etc.
- Upgrade insecure third-party dependencies. Please update all used by the application’s dependencies to prevent exploitation of issues that they introduce.

Long-term recommendations are:

- Implement regular security reviews practice. Regular vulnerability assessments will help you to mitigate security issues in the early stages and save your reputation and money.
- Add secure coding practices to the development process. Please provide training for developers to expand their knowledge in secure coding. It will help you to mitigate many vulnerabilities in the implementation phase.

Use package analyzers, static and dynamic application security testing tools in your development processes. Integrate tools such as the Snyk, Burp Suite, SonarQube, OWASP ZAP that will help you to detect common issues and vulnerabilities in the application.

Vulnerability report

In this section you will find a detailed description of all the detected vulnerabilities, together with a step by step instruction on how to reproduce every vulnerability and evidence of a successful exploitation. But, before you move on to the vulnerabilities, here is a brief explanation of how to understand unique identifiers of the documented issues. We use the following identifier template for every issue:

JSA-1.2

where,

- ☐ JSA (Juice Shop Application) - abbreviation that describes the tested application
- ☐ 1 - numeric identifier for severity level of the detected vulnerability. Here we mark severity levels according to the priority of mitigation measures for that issues, so the most critical issue will receive lower number

Severity	Numeric identifier (Priority)
Critical	1
High	2
Moderate	3
Low	4
Informational	5

- ☐ 2 - numeric identifier for a specific vulnerability in the selected severity level

demo.owasp-juice.shop

JSA-2.1. Unprotected FTP server

General information

Description	Using spiders we detected an unprotected FTP server that leaks sensitive application files.
CVSS 3.1 score	**7.5 High**
References to classifications	A01:2021
Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

Web application domain for scanning was provided by the customer before the penetration test started. It's publicly available information and can be found in OWASP juice-shop project official documentation. We conduct web application vulnerability assessment using OWASP ZAP scanner.

During the ZAP scan we detected FTP server that contained confidential information (see fig. 4).

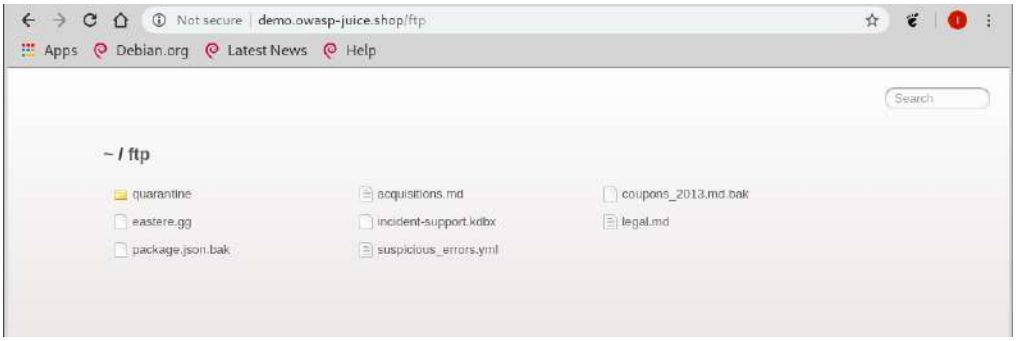


Figure 2.1.1. FTP folder with confidential data.

JSA-2.2. Retrieve backup files using null byte injection

General information

Description	An attacker is able to read sensitive files on the application server using null byte injection.
CVSS 3.1 score	**7.5 High**
References to classifications	A01:2021
Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

We found two backup files and tried to open them but received 403 Error from the server.

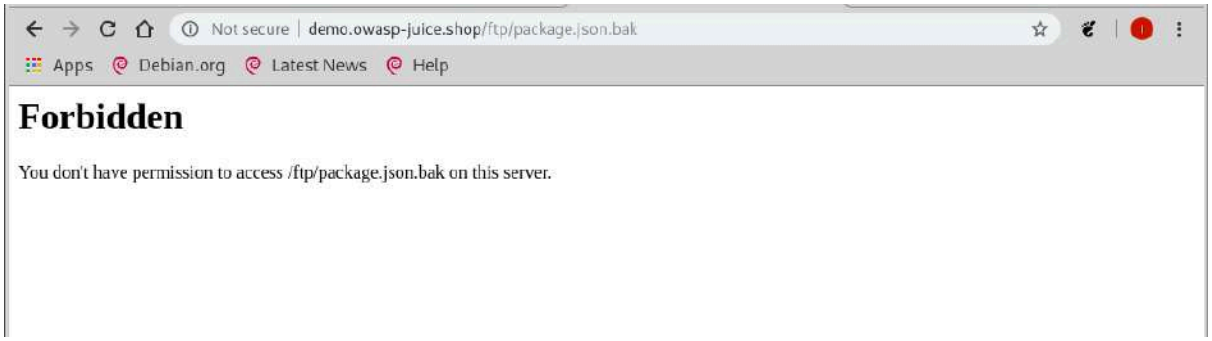


Figure 2.2.1. Attempt to open package.json.bak file.

During the investigation we found that server allows opening only '*.pdf' and '*.md' files. After that, we decided to apply null byte injection and tried to open stored on FTP server backup files. The attack was successfully conducted and we managed to download backup files from OWASP juice-shop ftp server.

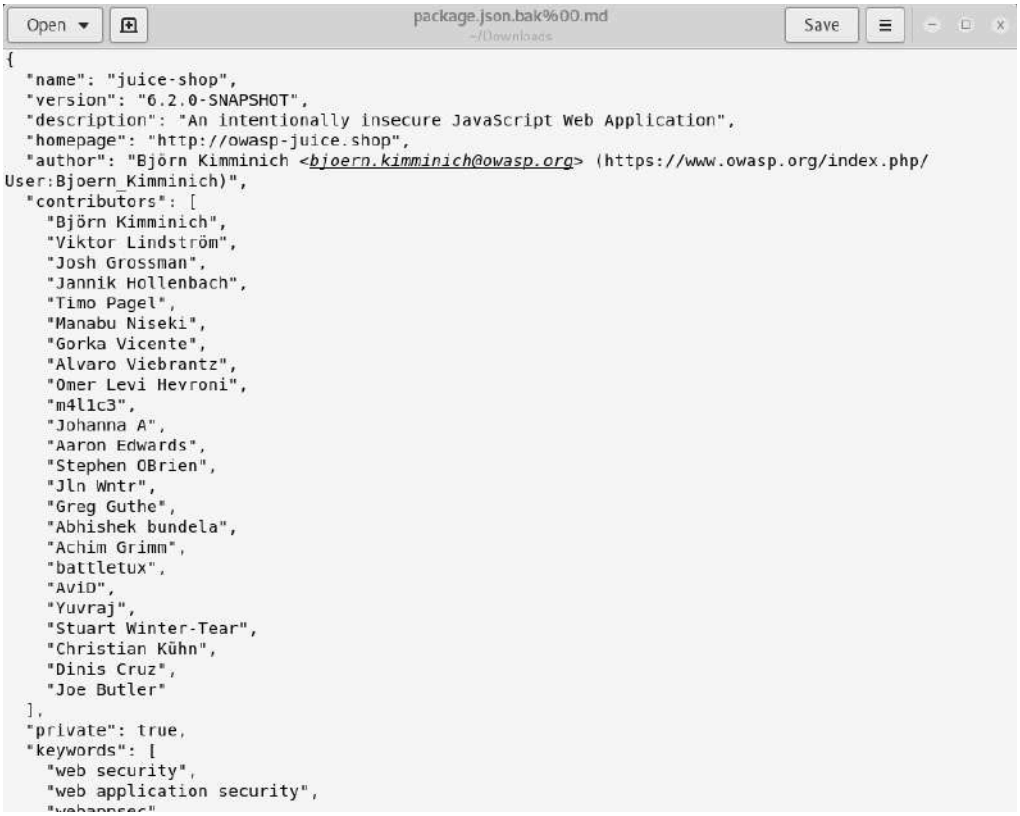


Figure 2.2.2. Downloaded using null byte injection package.json.bak file.

The file contains a full list of project dependencies and can be used for further investigation of potential vulnerabilities using Snyk dependencies scanner. Full package.json.bak file provided as an attachment.

JSA-4.1. Private IP disclosure

General information

Description	Automated scanners have detected misconfigured headers that can be used by attackers to attack an application, retrieve sensitive information about technologies that it uses, .etc.

CVSS 3.1 score	**3.9 Low**
References to classifications	A05:2021
Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

ZAP found private IP disclosure vulnerability during the initial scan. This information might be helpful for further attacks targeting internal systems. Also, the file contains the URL list with other application domains that could be investigated in the next stages in order to reveal confidential information and find new vulnerabilities. Complete ZAP scan report provided as an attachment.

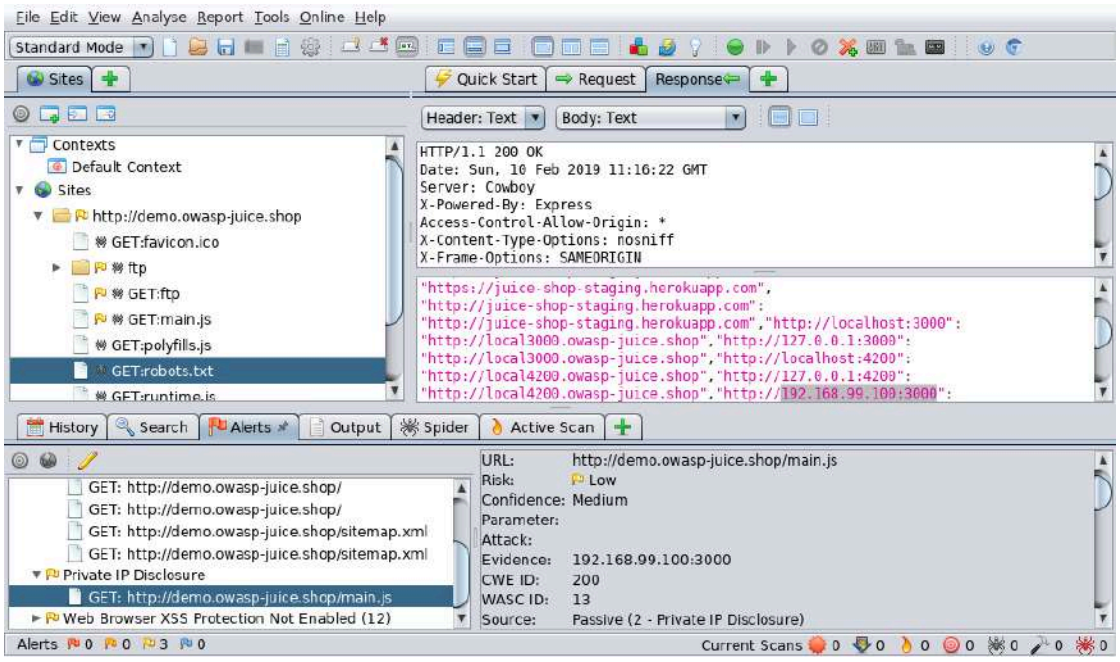


Figure 2.3.1. Private IP disclosure.

JSA-2.3. Bypass authentication using SQLi

General information

Description	We detected a SQL injection vulnerability that allowed us to bypass an authentication mechanism and get access to any account in the system.
CVSS 3.1 score	**8.2 High**
References to classifications	A03:2021
Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

We conducted penetration testing of the login form and revealed flow to execute SQLi injection attack against application authentication and get access to another user account, including administrator account. It was detected due to insufficient user input validation and broken server-side error handling. The first step was to provoke internal server error using invalid data. It allowed us to provoke internal server error which resulted in source code disclosure with a complete request to the database during login operation.

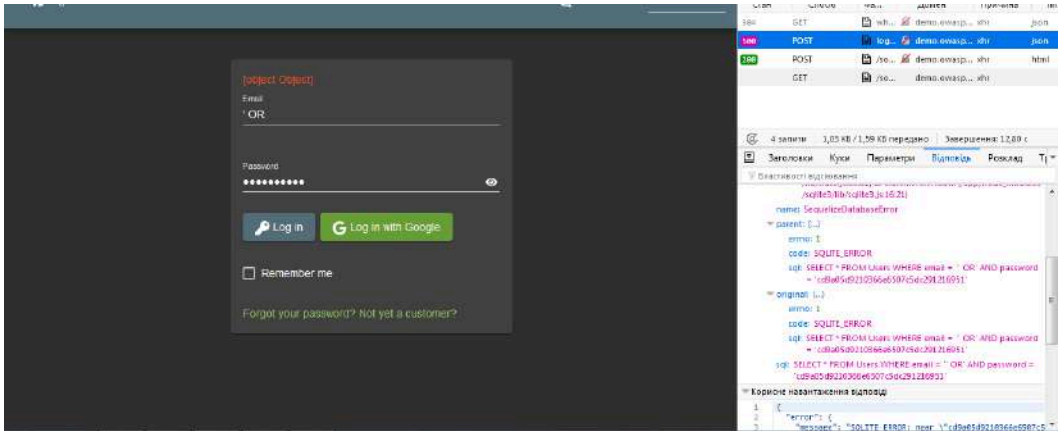


Figure 2.4.1. Internal Server Error that provides full SQL query in response.

We decided to conduct a SQLi attack after a detailed server error response analysis. During the attack, we managed to log into the application under an admin account using SQLi injection.

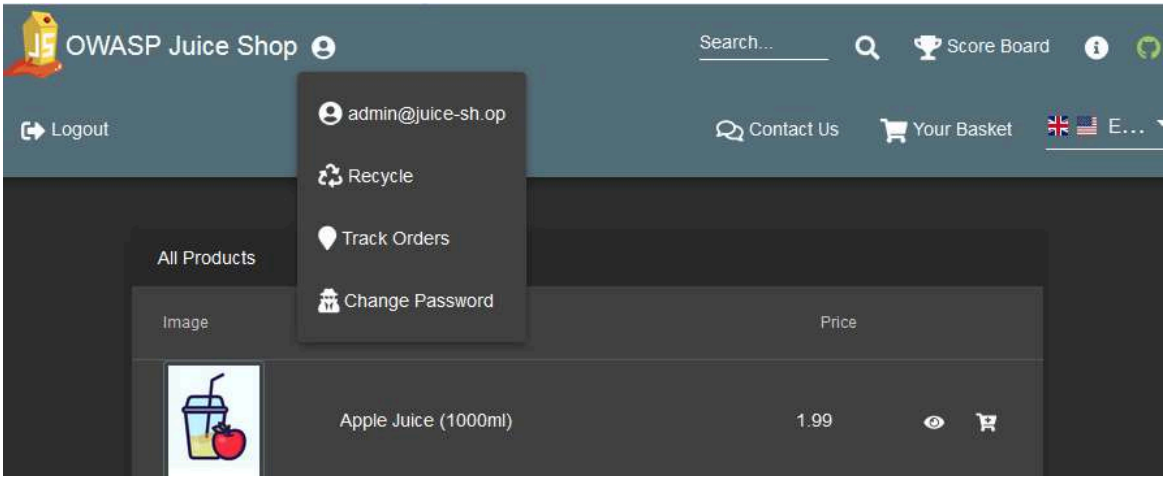


Figure 2.4.2. Successful authentication bypass using SQLi.

JSA-3.1. Reflected XSS in search functionality

General information

Description	Automated scanners have detected reflected XSS vulnerability that allows a malicious actor to execute arbitrary JS scripts, .etc.
CVSS 3.1 score	**5.3 Moderate**
References to classifications	A03:2021, CWE-79
Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

Our next target was search functionality. Using common templates and best practices we managed to conduct reflected XSS attacks. The attack resulted in an alert that contained user cookies. Further investigation of this threat vector could allow the attacker to steal user credentials or redirect him to the malicious site using spear-phishing technique.

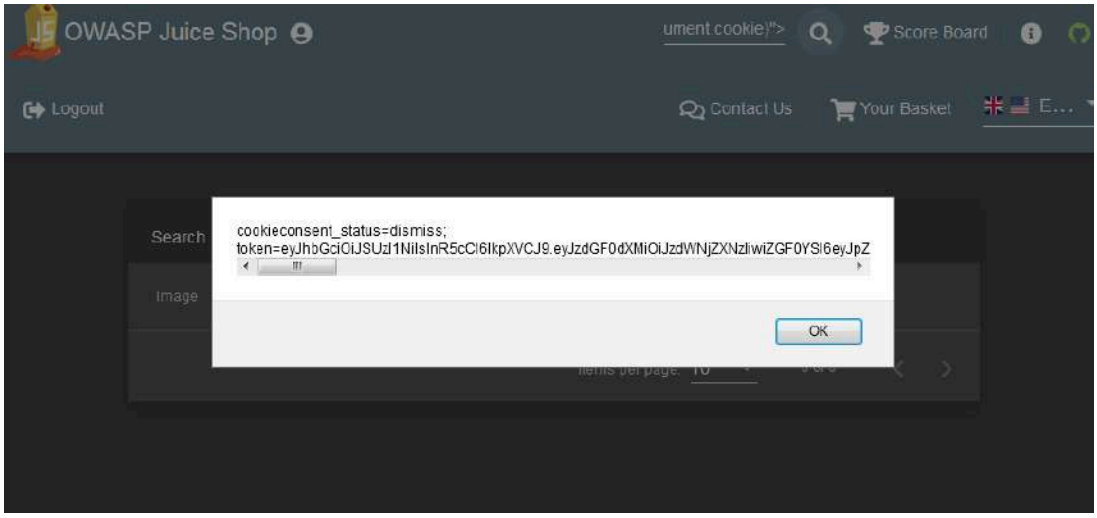


Figure 2.5.1. Successfully conducted reflected XSS attack

JSA-3.2. Broken access control on product reviews

General information

Description	Broken Access Control issue in product reviews allows attackers to submit comments on behalf of any application user.
CVSS 3.1 score	**5.3 Moderate**
References to classifications	A01:2021

Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

To identify broken access control we used Burp Suite proxy interceptor and modified application requests. The first step was to retrieve the id of the target user from the reviewers list, returned from the server. After that, we posted a comment under the admin account and using interceptor replaced author email in the request payload.

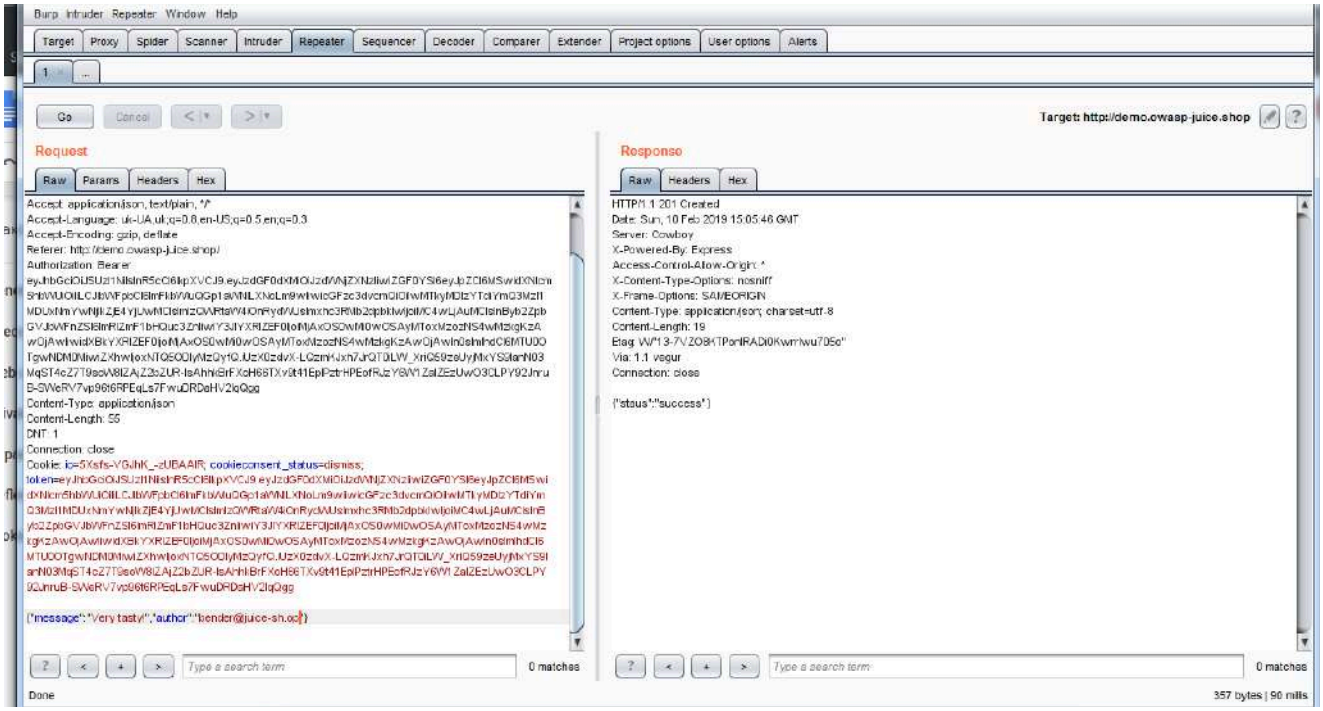


Figure 2.6.1. Post comment on behalf of another user.

Figure 2.6.2. Post comment on behalf of another user.

JSA-2.4. Arbitrary file upload

General information

Description	We detected a vulnerability that allows an attacker to bypass file upload restrictions and upload a file with arbitrary extension to the application server.
CVSS 3.1 score	**8.2 High**
References to classifications	A04:2021, CWE-434
Status	**Fixed. Date of retest: dd.mm.yyyy**

Exploitation proof of concept

We discovered a form for complaint where the customer is able to download the file and submit it along with his complaint to the application server. Using developer tools console we changed accepted type of input element to '*.xml' and downloaded maliciously crafted XML in order to submit it to the server.

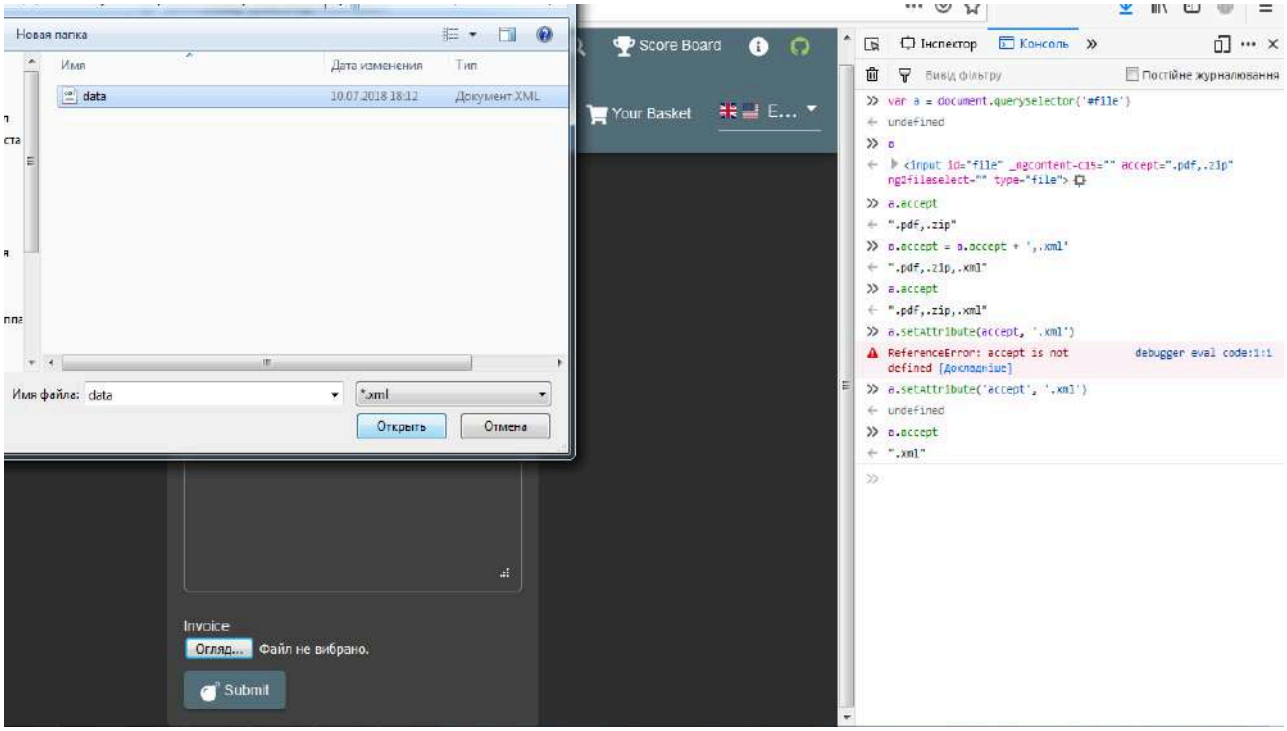


Figure 2.7.1. File type restrictions bypass.

After the file was downloaded to the application server it could be executed and would lead to server compromise.

Remediation report

Recommendations

Due to the impact of disclosed vulnerabilities, appropriate resources should be allocated to mitigate risks and protect the application from malicious threat actors. Future software development processes should be conducted according to SSDLC's (Secure Software Development Lifecycle) best practices to prevent additional attack vectors. The complete list of SSDLC recommendations is listed in OWASP Secure SDLC [4].

We recommend the following:

1. **Fix broken access control vulnerabilities.** Please add verification of user permissions on the server-side to mitigate broken access control issues because it leads to user impersonation. An attacker could submit malicious data on behalf of another application user or access his private data. For example, a threat actor could submit a comment on behalf of another user. Please adjust those few issues where the access matrix is not followed.
2. **Fix file upload issues.** Add more checks to prevent uploading of invalid files to the application server. It could lead to remote command execution, sensitive information disclosure and server crashes.
3. **Add strong server-side validation.** Because of the high risk of client-side validation bypass, there should be implemented strong server-side validation. It could mitigate such issues as stored XSS, SQLi, arbitrary file upload attacks. In the case of arbitrary file upload please define a list of acceptable extensions on the server-side and always perform validation of user input.
4. **Add client-side validation.** Strong client-side validation could mitigate reflected XSS attacks. It's important due to the consequences of this attack vector. Using unprotected application functionality attackers could use it to steal users' credentials, share malware and other malicious actions. We recommend following security best practices for MEAN stack projects (as applications implemented using

Angular on the client-side) listed here: https://angular.io/guide/security and adding encoding to the client-side as to the server-side validation.

5. **Implement regular security reviews practice**. Regular vulnerabilities assessments would allow you to mitigate security issues in the early stages and save your reputation and money.
6. **Add secure coding practices to the development process.** Please provide training for developers to expand their knowledge in secure coding. It will help you to mitigate many vulnerabilities in the implementation phase.
7. **Use packages analyzer.** Integrate Snyk (https://snyk.io/) tool into your development process to prevent usage of components with known vulnerabilities. Attackers could use them to conduct successful attacks against apps.
8. **Keep software up to date.** Outdated versions carry vulnerabilities and can be dangerous for the application.

Library

1. OWASP Top Ten project: online resource. - Access link: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

2. PTES (Penetration Testing Execution Standard): online resource. - Access link: http://www.pentest-standard.org/index.php/Main_Page

3. OWASP Testing Checklist: online resource. - Access link: https://github.com/tanprathan/OWASP-Testing-Checklist

4. Secure SDLC: online resource. - Access link: https://www.owasp.org/index.php/Secure_SDLC_Cheat_Sheet