

The Secret Sanctum Walkthrough

Task 2

Hunt me!!

Task 3

The Secret Sanctum

How well can you exploit a superhero's web application with weak security and prove yourself to be a superhero? head to <http://164.92.196.115/>

Answer the questions below

Flag 1

Answer format: ****{*****}

Submit

Hint

Flag 2




Answer format: ****{*****}

Submit








Flag 3

Answer format: ****{*****}

Submit

Created by	Room Type	Users in Room	Created
 th3madbit  wambuapeter  Masara	Free Room. Anyone can deploy virtual machines in the room (without being subscribed)!	17	4 days ago

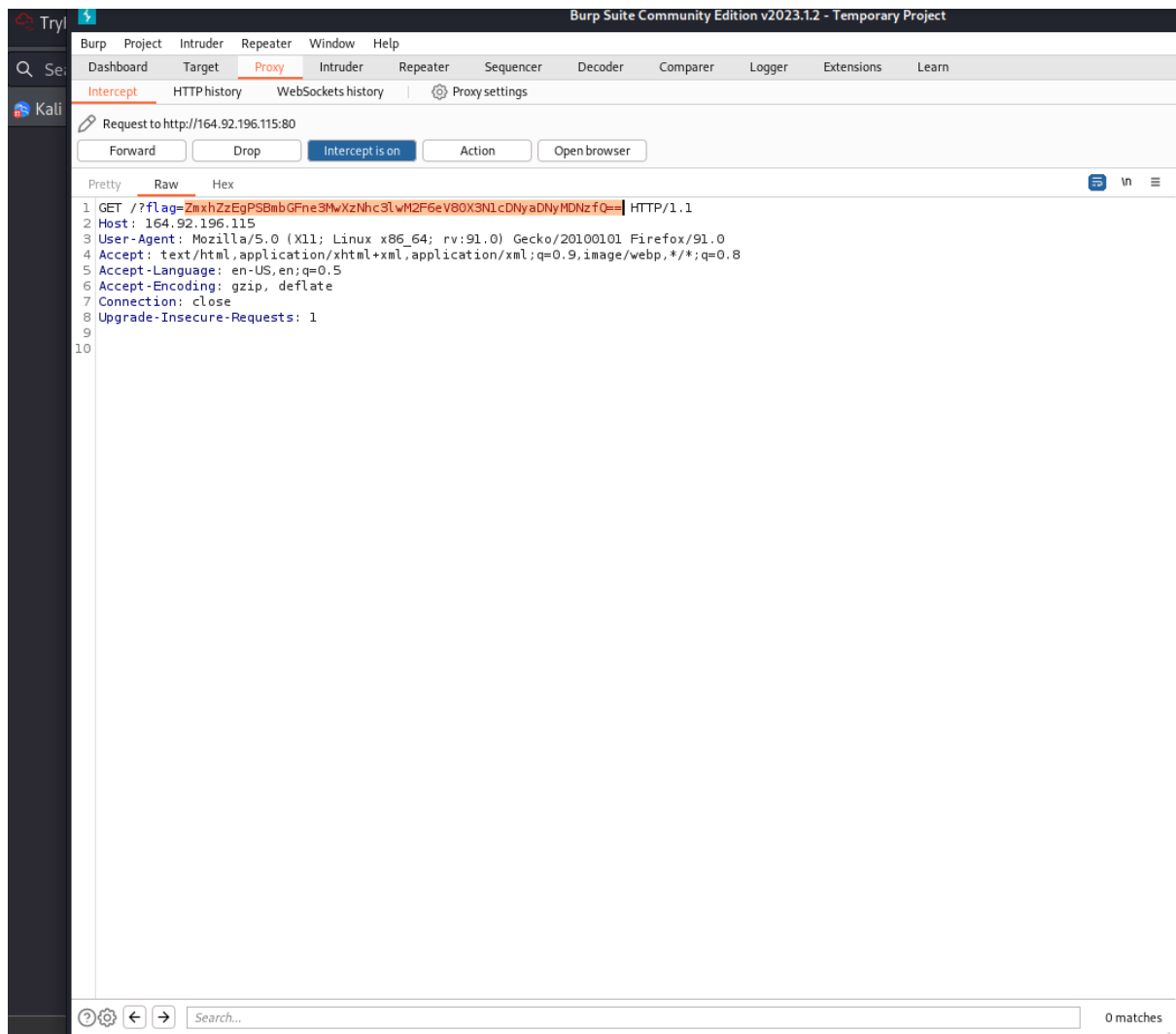
Copyright TryHackMe 2018-2024



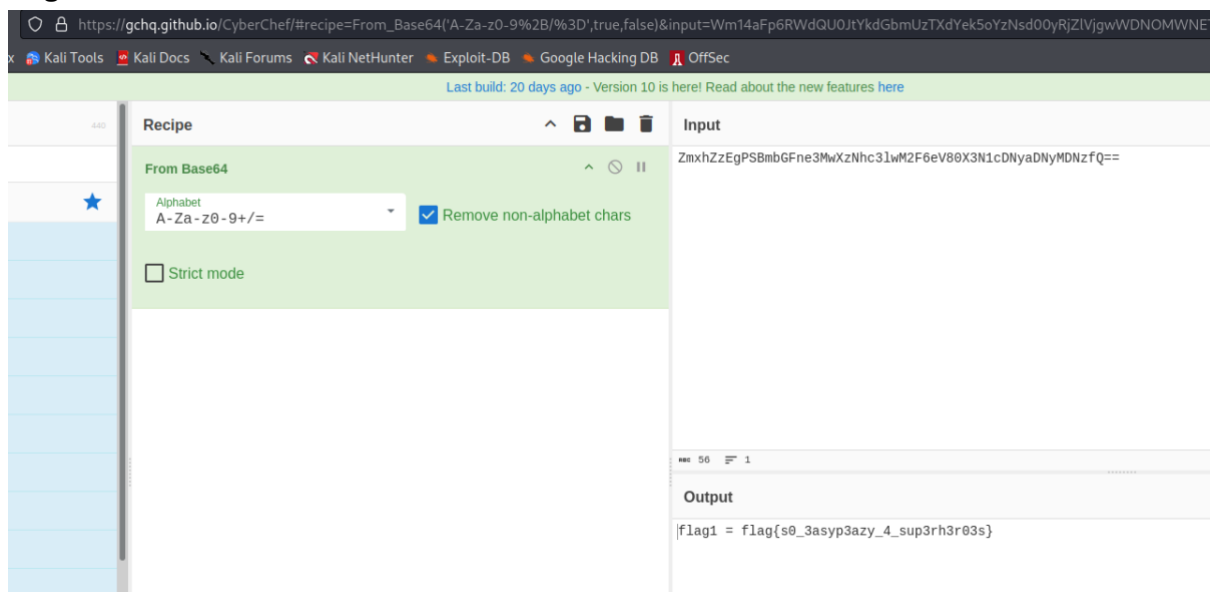
This is just a simple challenge I created for the purpose of understanding web applications and some of the most popular vulnerabilities used to attack web applications. This challenge needed understanding of url redirection, sql injections and Cross Site Scripting.

Visiting the url given, we realize that the application redirects us to another page. Though, in the redirection, their is a string passed. Intercepting this with burpsuite to inspect closely, we find it is a base64

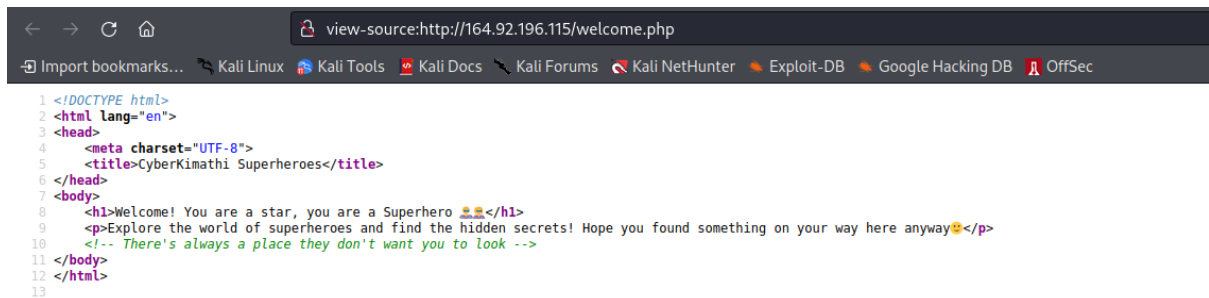
string:



Decoding it, we get the
flag

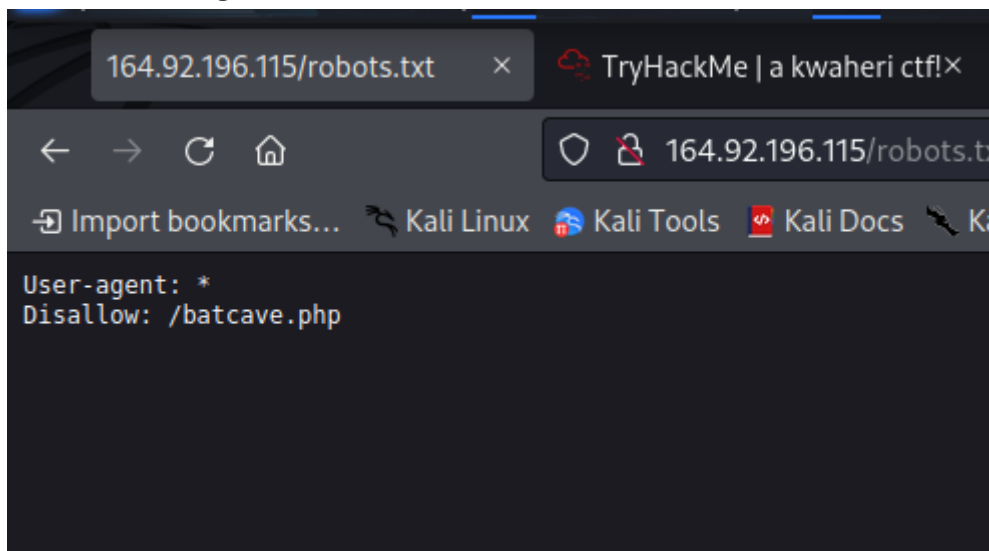


Proceeding, we find a page that tells us to explore the hidden secrets. I don't know about you but I first think of viewing the page source for comments after hearing this.



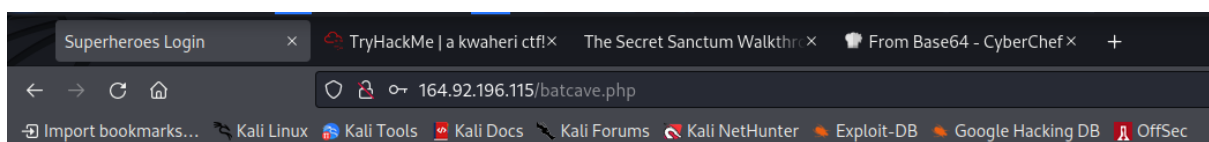
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>CyberKinathi Superheroes</title>
6 </head>
7 <body>
8   <h1>Welcome! You are a star, you are a Superhero 🌟</h1>
9   <p>Explore the world of superheroes and find the hidden secrets! Hope you found something on your way here anyway🌟</p>
10  <!-- There's always a place they don't want you to look -->
11 </body>
12 </html>
13
```

The comments say that there is always a place they don't want you to find. There is a page in web applications called robots.txt that states pages that they don't want indexed. Visiting it:



```
User-agent: *
Disallow: /batcave.php
```

So we visit the page /batcave.php. We are presented with a login page. Trying default creds like 'admin': 'admin' and 'admin': 'password' doesn't work and says incorrect credentials.



Login

Sorry but I don't care how powerful you are, you need to get past me first 🤖

Username:

Password:

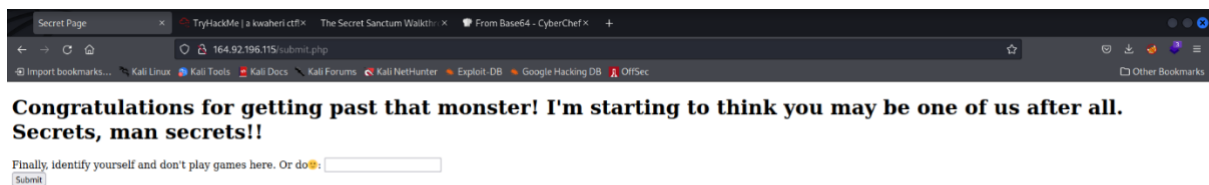
Invalid login credentials.

But then trying to invoke an error by adding a character to the parameters, like admin' for the username causes an error and the whole page disappears. This means that input is not being filtered and it is vulnerable to sql injection. Try this payload for username and anything for password:

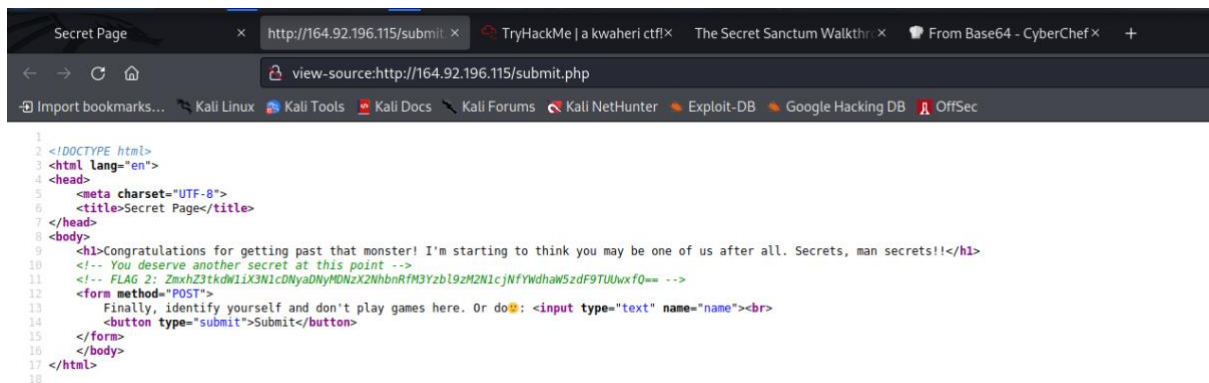
admin' OR 1=1;--

Explanation: the ' closes the parameter, then since 1 is equal to 1, this returns True. Now since the second part of the statement is true and we have used OR to mean if any part of the statement is true the whole is true, and commented everything else, it means no matter what we enter for the password, the login form returns true hence we are logged in. This log us in.

We find a page that insists for secrets.



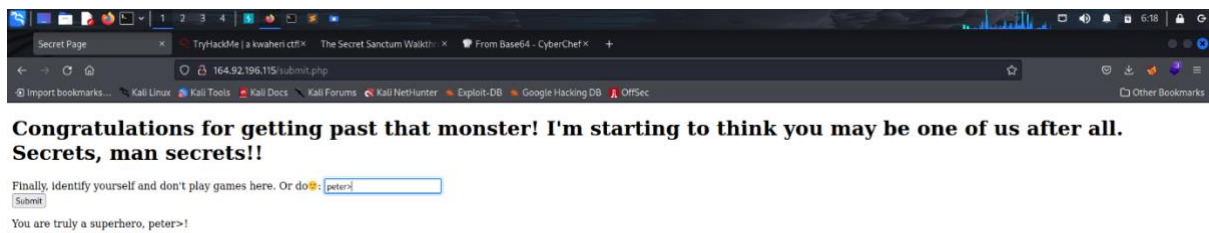
Viewing page source, we get a base64 for the second flag



Now looking at the functionality of the page we are on, it has a form that needs you to identify yourself to check if you are a superhero. Entering your name, it returns the statement You truly are a superhero, then your name.

Checking if it filters user input, enter something like "name>". It does not

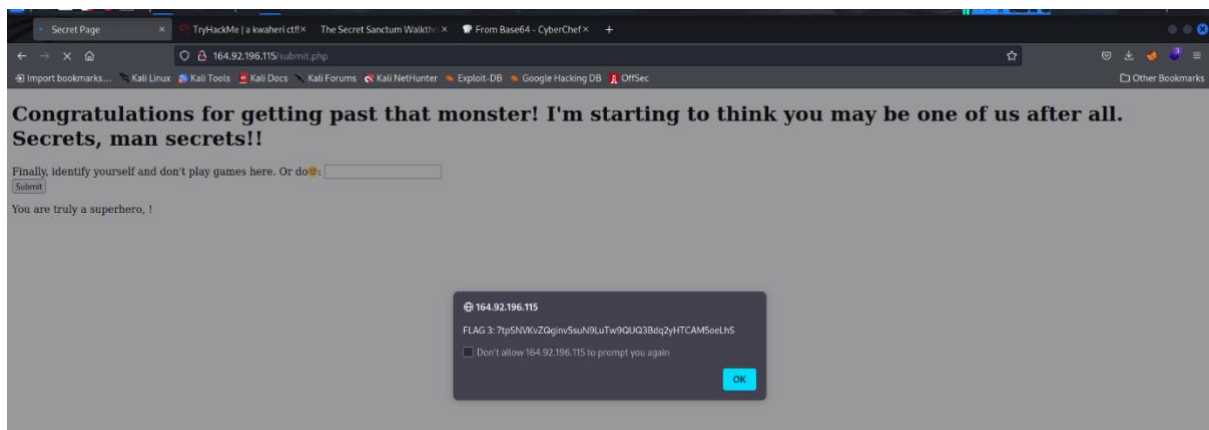
filter



That means the application will suffer from XSS due to this. Now use the payload

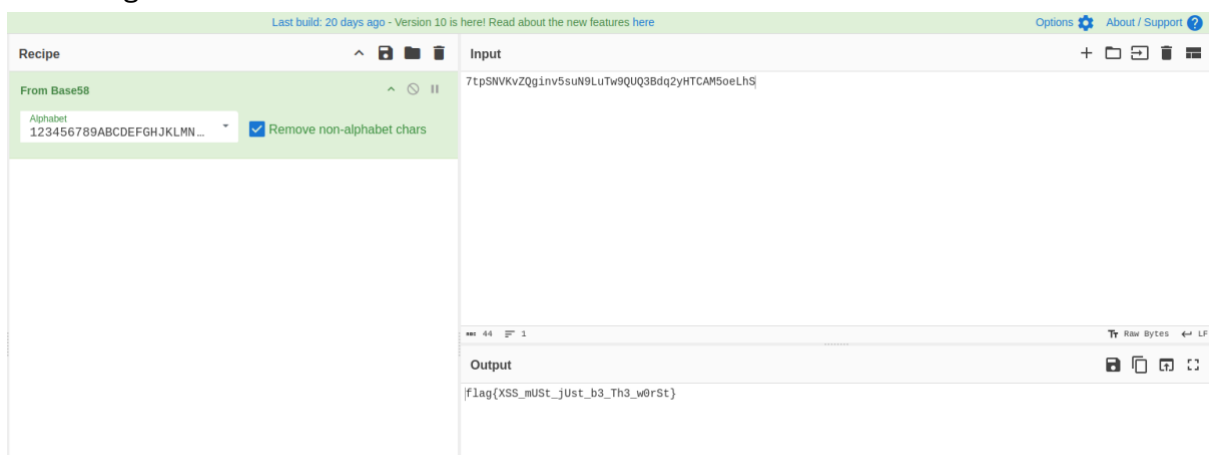
```
<script>alert(1);</script>
```

This throws an alert, then discloses the flag which is encoded.



Decoding this with [cyberchef](#), it is not base64. Use the magic operation, it discloses the string to be base58.

Decoding:



FUN isn't it?