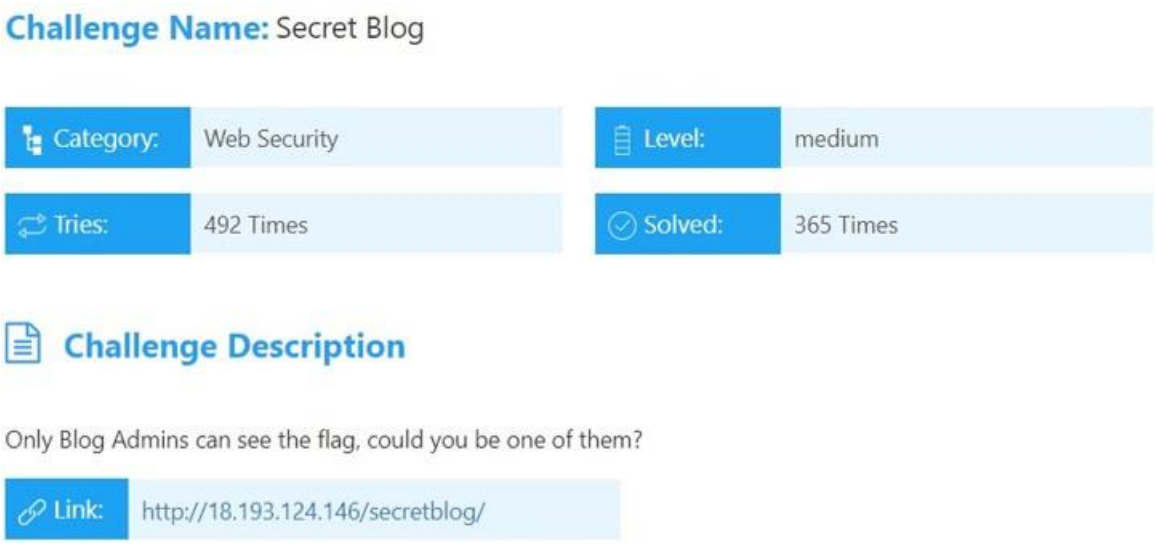
Author: Kharim Mchatta

Title: CYBER TALENTS CTF WRITE UP

Category: WEB

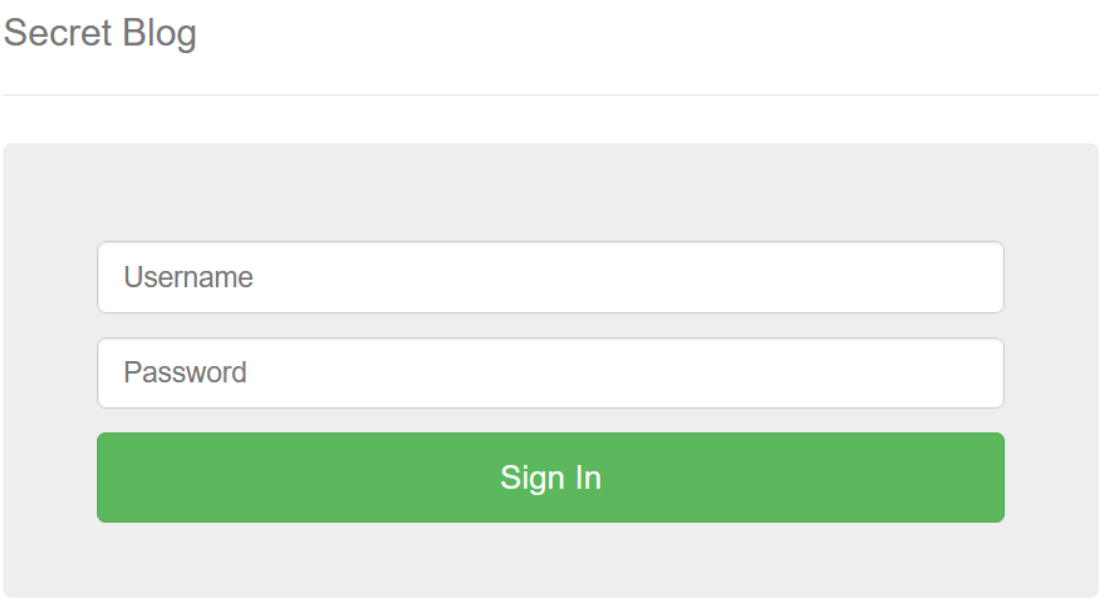
Date: 7/12/2021

1. SECRET BLOG



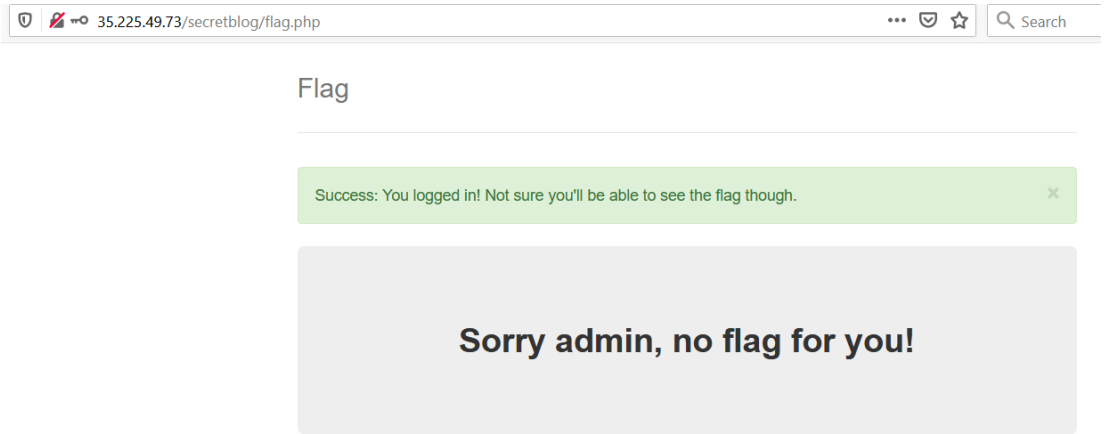
We start off with this challenge called secret blog, the description of the challenge states that **"only blog admin can see the flag, could you be one of them?"**

Clicking on the provided link we are redirected to a webpage which has a login.



I looked at the source code and there was nothing interesting there, I tried looking at robots.txt and the page returned a 404.

The next step was for me to try default credentials which included username and password: **admin** which logged be on the website and found the below message which said I had to be an admin to get the flag.



In order to get the flag we had to understand how the requests are being send from the user to the server, so I went back to the login page then inserted the username and password and then intercepted the request using burpsuite.

```
POST /secretblog/login.php HTTP/1.1
Host: 35.225.49.73
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101
Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://35.225.49.73
Connection: close
Referer: http://35.225.49.73/secretblog/
Cookie: username=admin; password=admin; admin=False
Upgrade-Insecure-Requests: 1

Username=admin&password=admin
```

Examining the request I had seen the username and password fields that we were sending to the server and examining further I had seen on the cookie that the admin was set to false

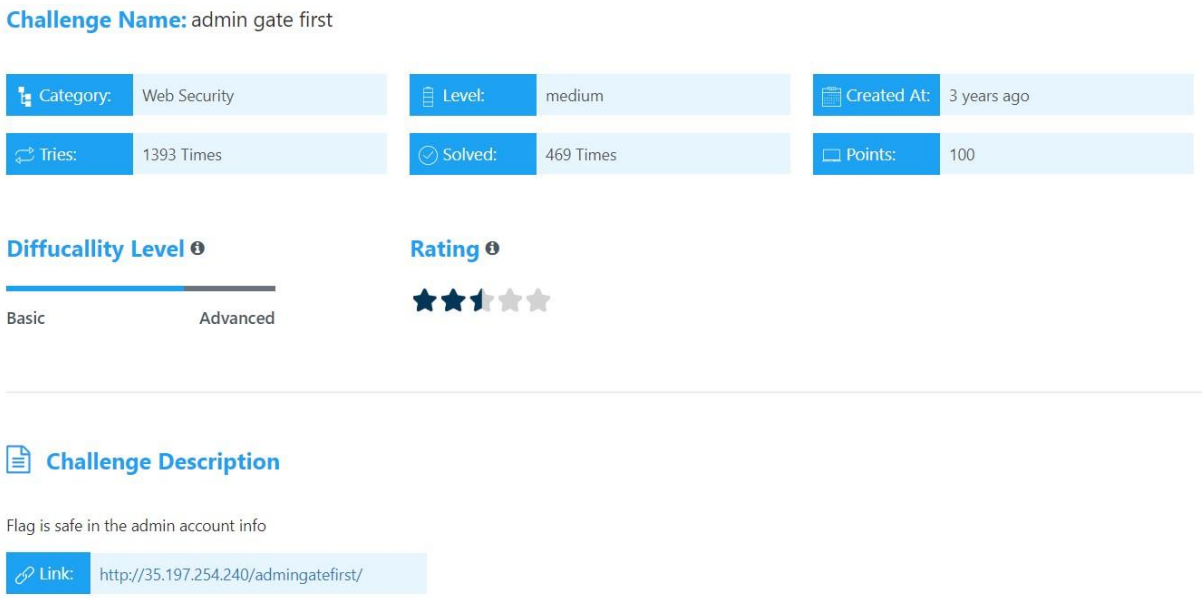
Changing the parameter of the cookie from admin=False; to admin=True; and forwarded the request to the server

```
GET /secretblog/flag.php HTTP/1.1
Host: 35.225.49.73
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101
Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://35.225.49.73/secretblog/
Connection: close
Cookie: username=admin; password=admin; admin=True
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

After checking out the out put as shown below we had obtained the flag.

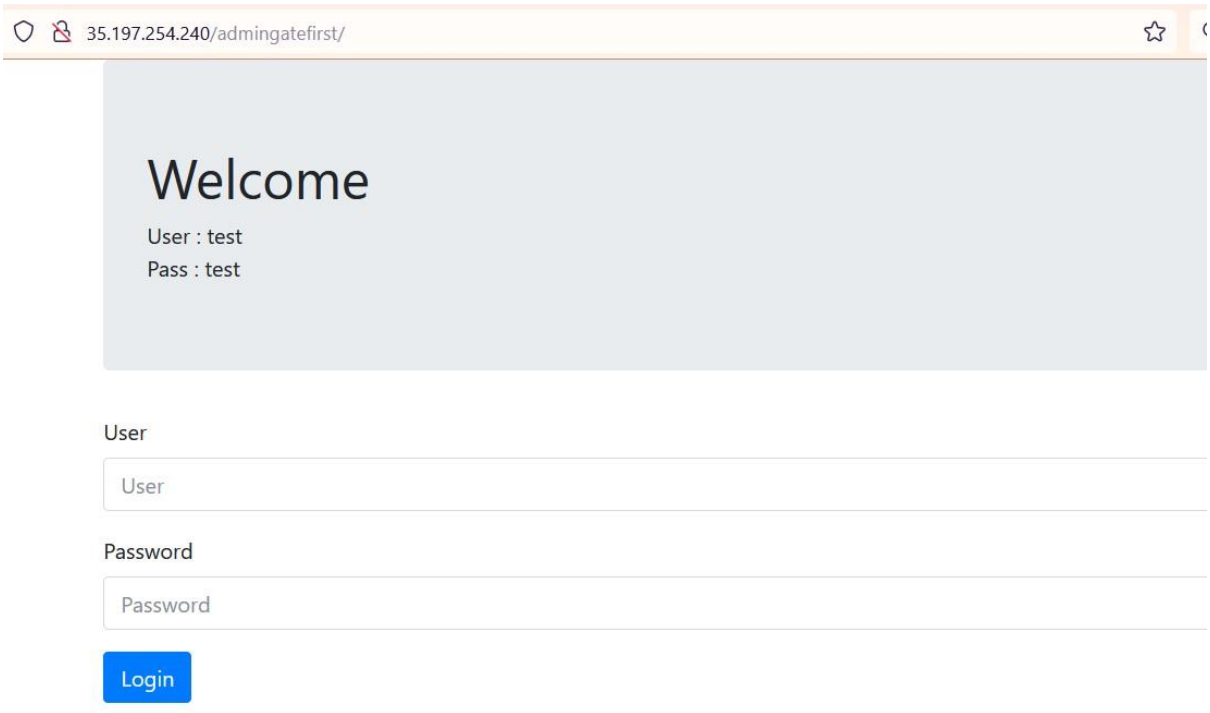
```
<div class="container">
  <div class="header">
    <h3 class="text-muted">
      Flag
    </h3>
  </div>


  <div class="alert alert-success alert-dismissible" role="alert" id="myAlert">
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
    Success: You logged in! Not sure you&#39;ll be able to see the flag though.
  </div>


  <div class="jumbotron">
    <p class="lead">
    </p>
    <p style="text-align:center; font-size:30px;">
      <b>
        flag{I_l0v3_CoOki3s_M4nipul4ti0n}
```

2. Admin Gate First

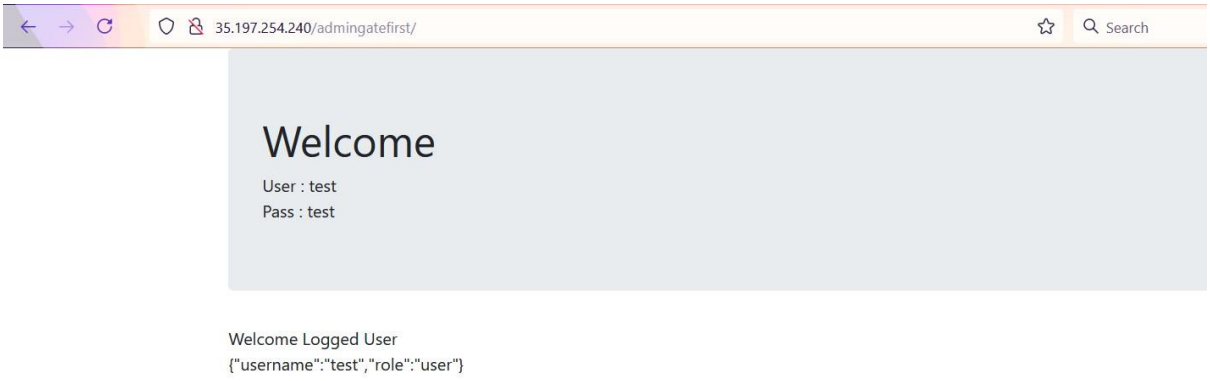


This challenge was called admin gate first. Its description states that **"the flag is safe in the admin account info"**, meaning that in order to access the flag we need to get to the account of the admin.

Opening the provided link we are greeted with a login page with some credentials which we can use to login the website.

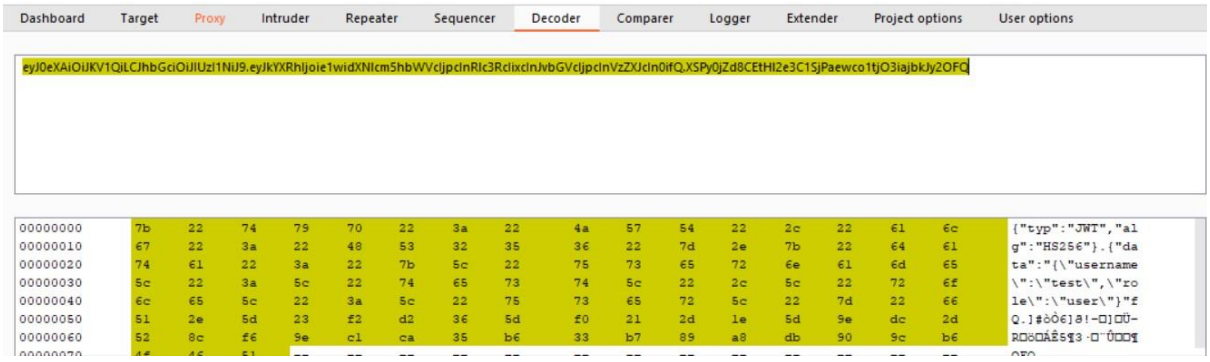


Login the website we see that the site says welcome and shows the user that logged in and its role. Now its up to use to find a way to change our role from user to admin.



Next, I had to intercept the traffic with burpsuite and see what I was dealing with. I logged out and logged back in with the credentials and intercepted the traffic and started to examine the request. I noticed under cookie there was an authentication that was been sent to the server every time a user tries to login as shown below

Taking the authentication code and inserted it on burpsuite decider and used base64 to decode the authentication code and we noticed that it was a JWT – Json Web Token as shown below.



JSON is an open standard used to share security information between two parties a client and a server. They are usually used for authorization, for more information on JWT just watch this YouTube video through the following link

https://youtu.be/7Q17ubqLfaM

I took the authentication key and pasted it on in an online decoder on the below link

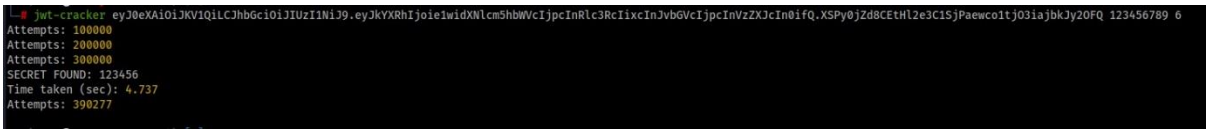
https://jwt.io/

and pasted the code and could see in a better view the content of the JWT.

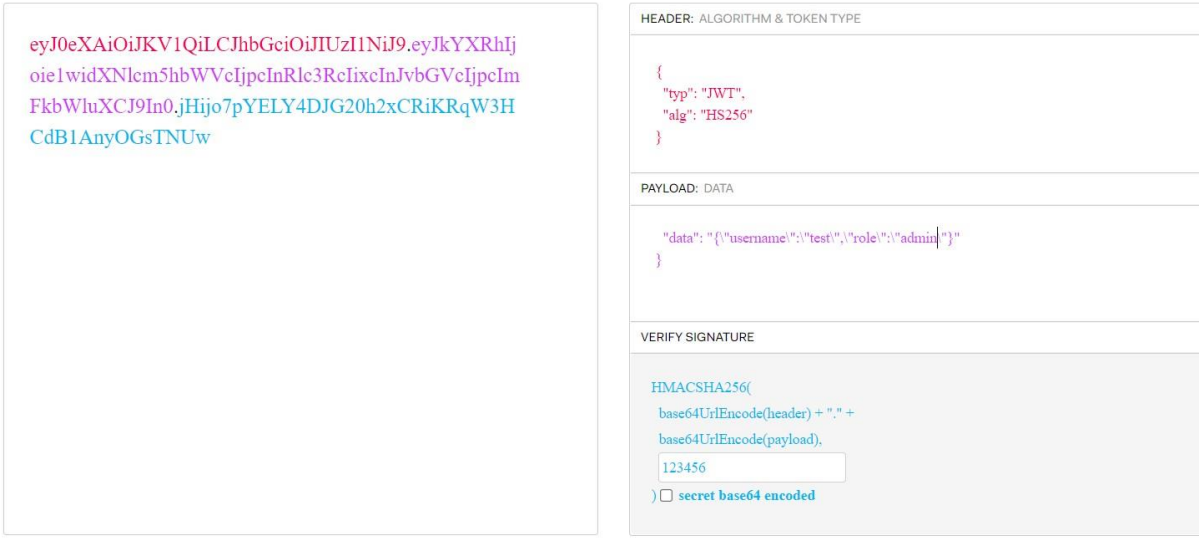
We could see that it the authentication was for the user test who had the role of user in the system. Next step is to try and crack the JWT token and then get the secret key.

I made use of a tool called jwt-cracker (https://lmammino.github.io/jwt-cracker/) but you could also use a tool from github called jwtcat (https://github.com/AresS31/jwtcat)

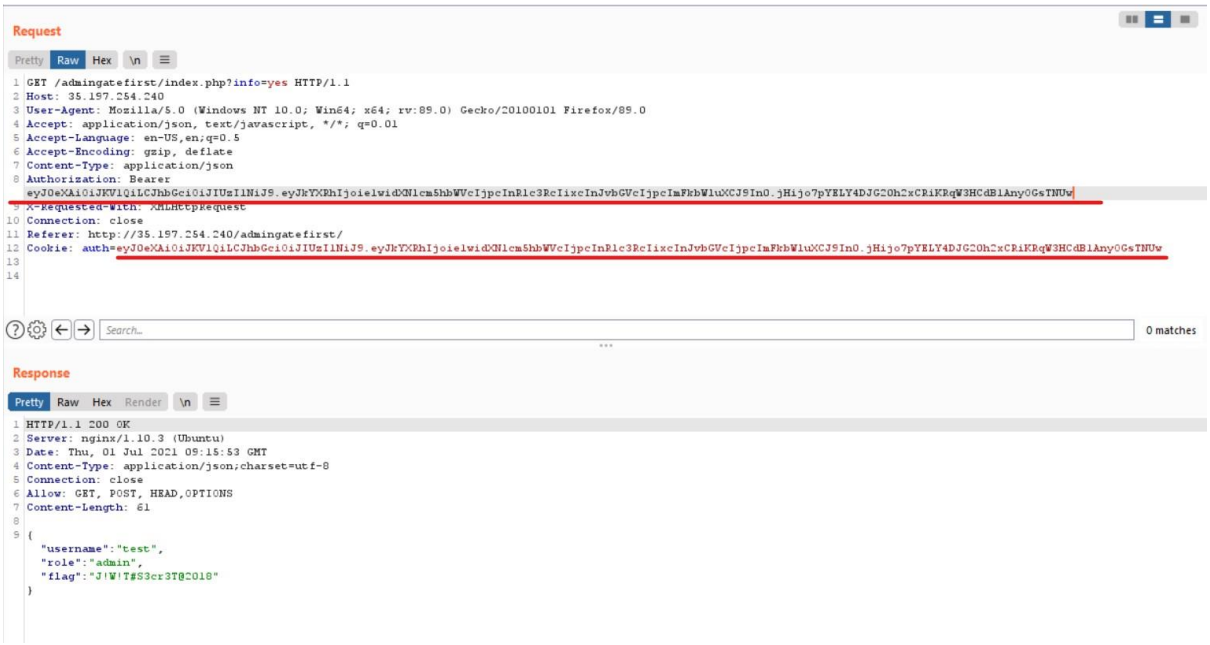
Cracking the token and we get the key which is **123456**



The next step would be changing the role to admin and inserting the key in the verify signature section. Once those changes are made you can copy the new JWT token ready for use.

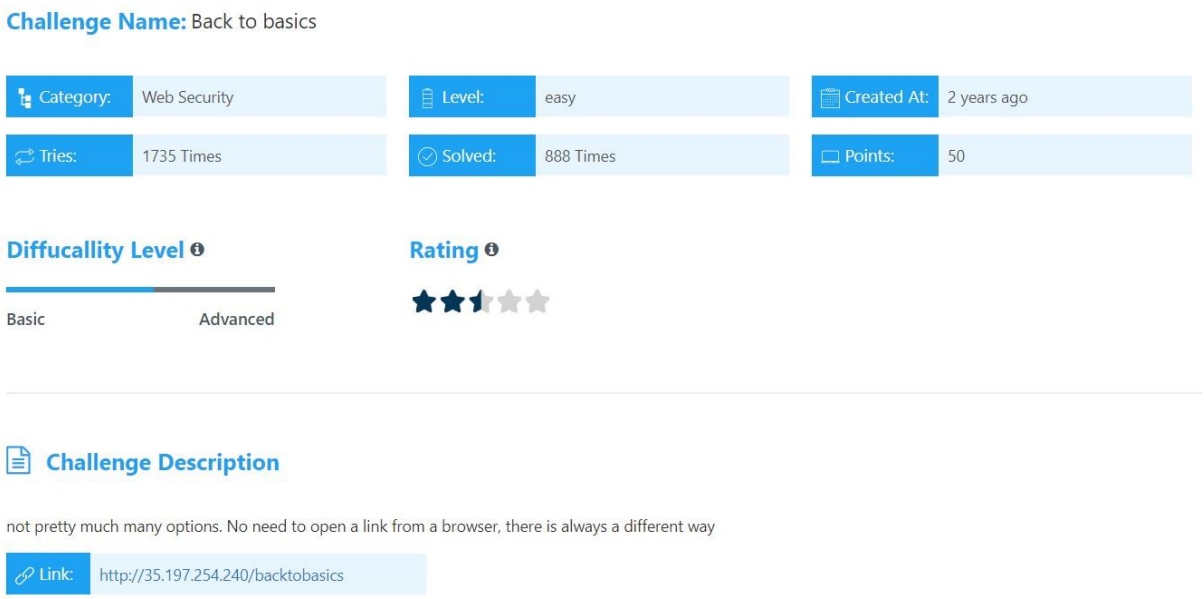


Going back to burpsuite we need to replace all the places which had the old JWT tokens with the new one which has the modified role from user to admin. This can be seen on the below image.



Once the request was submitted, we received our flag back.

3. Back to basics



This challenge was called back to basics, it was a tricky challenge but at the end of the day it was concurred. The challenge description was **"not pretty much many options. No need to open a link from a browser. There is always a different way"** as the description of the challenge states no need to open a link from the browser well, they really meant it and you shall know why shortly.

I took the link and pasted it on the browser and whenever I pasted it, it was redirecting me back to google page. I was initially baffled with this but then I decided to send the request to burpsuite and see what the reason of such behavior is. After intercepting the traffic I had sent it to repeater to see what was happening.

```
GET /backtobasics/ HTTP/1.1
Host: 35.197.254.240
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:84.0) Gecko/20100101 Firefox/84.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

After sending the request I had seen that the website was allowing different kind of HTTP Methods like GET, POST, HEAD and OPTIONS as shown below, and as for our bafflement on why we were been redirected to google is because of the script as seen below.

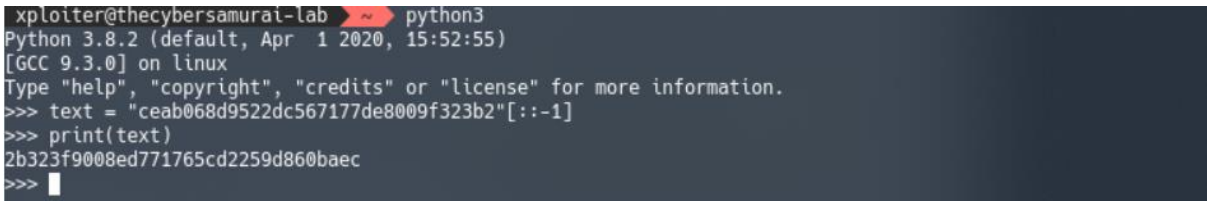
```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Tue, 29 Dec 2020 09:32:02 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Allow: GET, POST, HEAD,OPTIONS
Content-Length: 65

<script>
   document.location = "http://www.google.com";
</script>
```

Changing the Method from GET to POST and send the request to the server we can see in the response space there is a obfuscated code.

Took the code to java beautifier (https://beautifier.io/) so that the code can be readable.

Reading the content of the obfuscated flag we notice that the code requires us to reverse the string inside the 4th index in _0x6e5x2 variable. To reverse it we had to write a script as shown below.

Running the code we get the flag as show in the below image **2b323f9008ed771765cd2259d860baec**



4. Bean

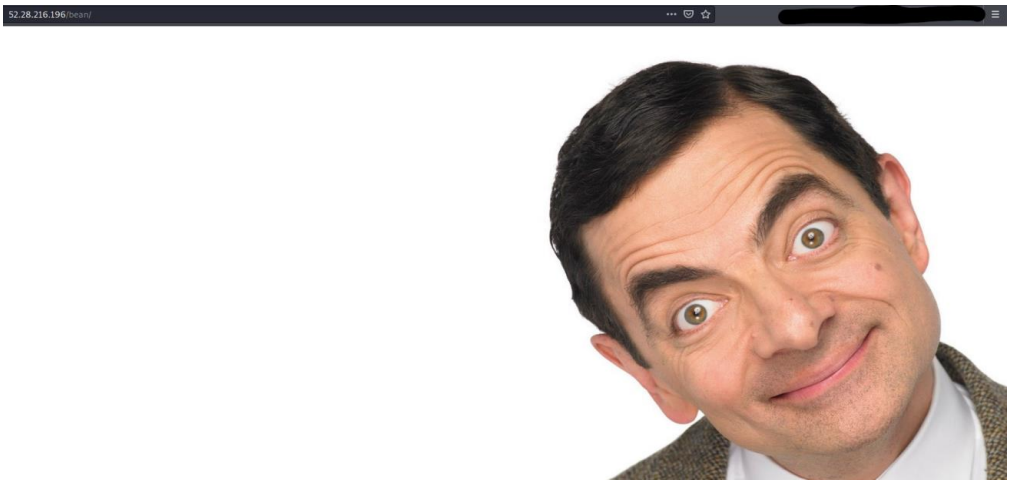
Challenge Name: bean

Category:	Web Security	Level:	easy
Tries:	2060 Times	Solved:	988 Times

Challenge Description

Challenge Link: http://a527d14de858a4a2ab03e28cc2870f07-751037772.us-west-1.elb.amazonaws.com

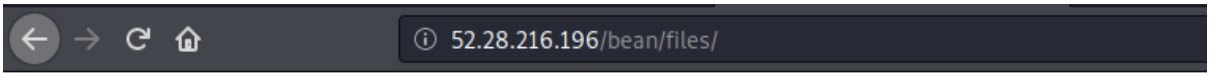
This challenge was called bean where the description said, **"come back home Mr Bean".** Opening up the link we just find a website with the image of Mr. bean, I checked robots.txt and inspected element but it had nothing useful.



Next, I went to brute force directory of the website to look for useful directories and found one path which was /files.

```
┌──(root💀Thecybersamurai)-[~/Desktop/cybertalents/bean]
└─# gobuster dir -u http://52.28.216.196/bean/ -w /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-medium.txt
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://52.28.216.196/bean/
[+] Threads:        10
[+] Wordlist:       /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2021/04/05 13:37:05 Starting gobuster
===============================================================
/files (Status: 301)
Progress: 7847 / 207644 (3.78%)
```

Then I went to the file path and found the server directories



Index of /files/

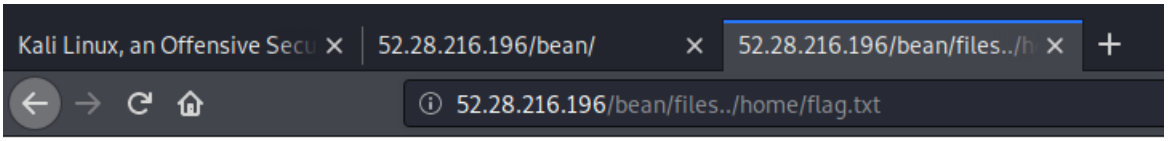
../		
alternatives/	04-Mar-2021 18:20	-
apt/	25-Apr-2017 17:19	-
cron.daily/	24-Apr-2017 17:03	-
default/	25-Apr-2017 17:20	-

Going back one step behind we find a home directory, going into the home directory we see the flag

Index of /files../home/

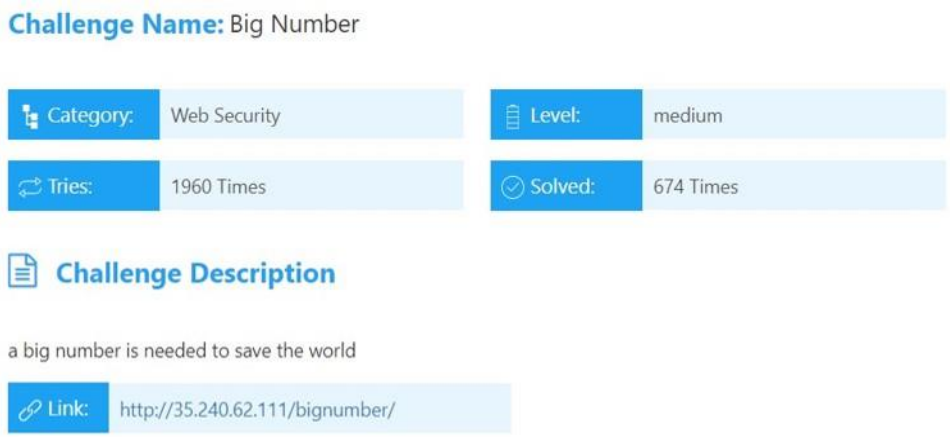
../		
flag.txt	18-Feb-2021 12:29	40

Viewing the content of the flag as shown below.

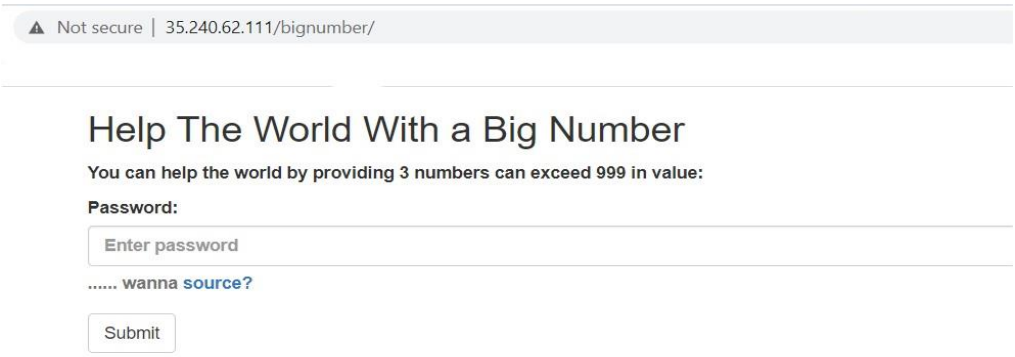



```
FLAG{Nginx_nOt_aLWays_sEcUre_bY_The_waY}
```

5. Big number



This challenge was called big number where the challenge description says that **"a big number is needed to save the world"** opening the provided link it led me to a webpage which asked for a password, but the password should be 3 numbers long and can exceed 999 in value as shown below



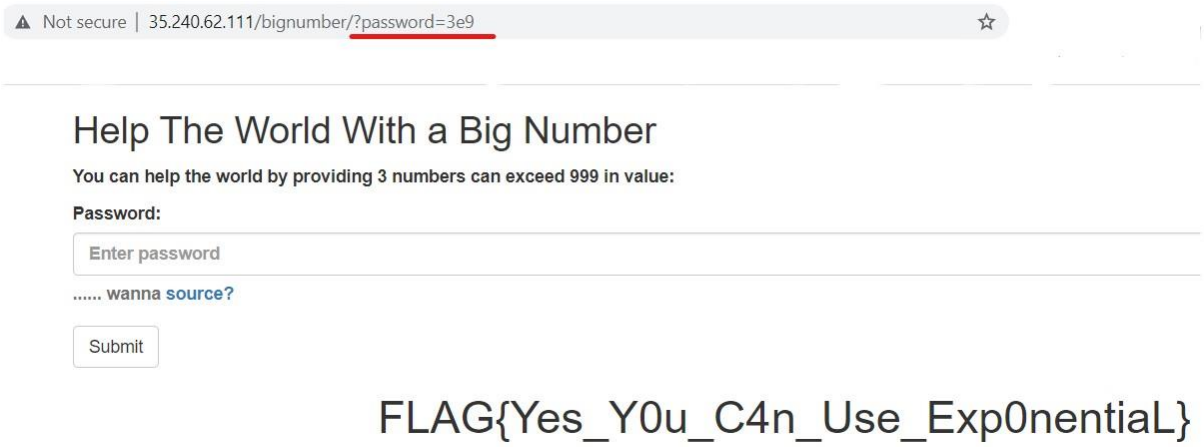
Clicking on the source tab a source code appeared which gave us a hint, as shown below.

```
<?php if (isset($_GET['password'])) {
    if (is_numeric($_GET['password'])){
        if (strlen($_GET['password']) < 4){
            if ($_GET['password'] > 999)
                die($flag);
            else
                print 'Too little';
        } else
                print 'Too long';
    } else
        print 'Password is not numeric';
}
?>
```

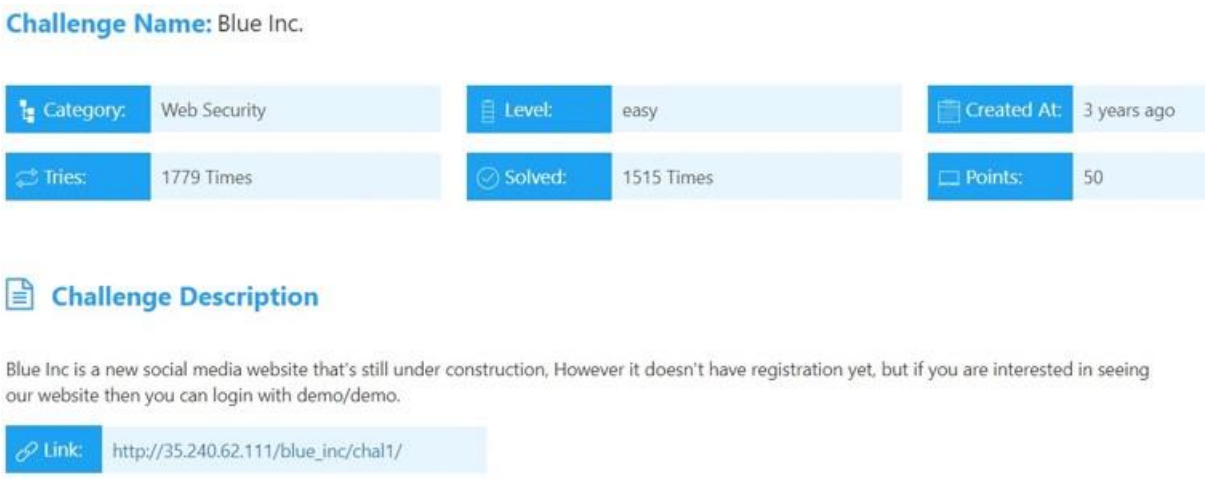
Basically, what the code is trying to highlight is we need a number where the number is less than 4 and also more than 999 in value. Logically its impossible to enter a number greater than 999 which is not 4 digits long and the only solution to this is exponentials in mathematics.

Exponential can allow us to make the value of 1 digit number increase, we need an exponential number that can give us a value more than 999 which is 3 exponential 9 but submitting this as it is would not work so we need to type its as 3e9 as the password.

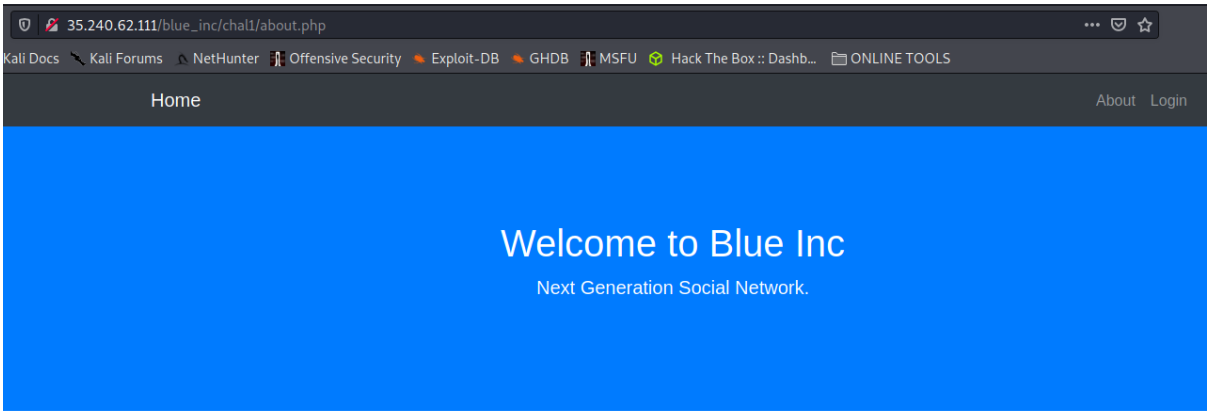
Submitting the value and I get the flag



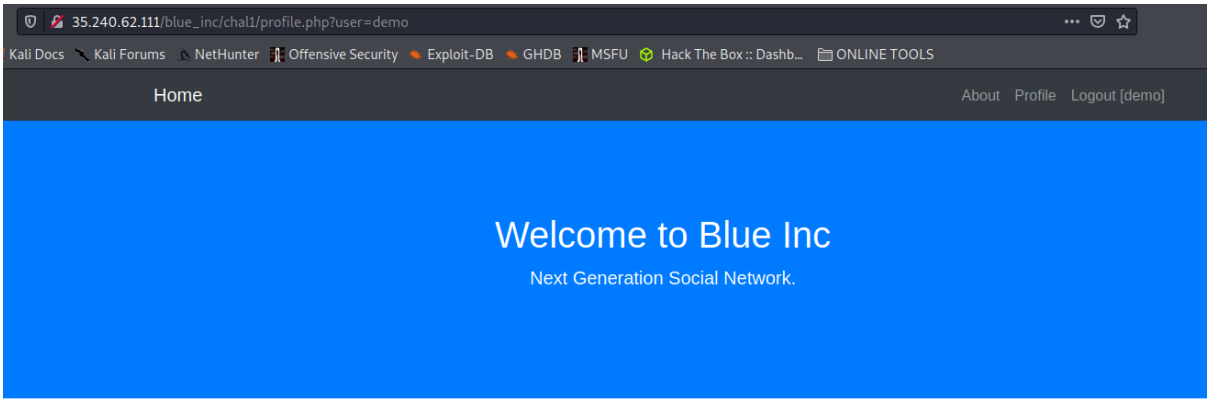
6. Blue Inc

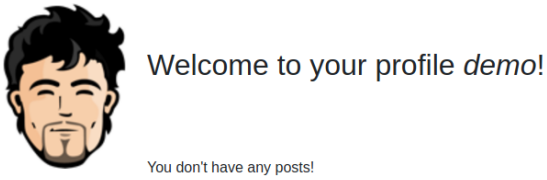


This challenge was called blue Inc where the challenge description says that **"Blue inc is a new social media website that is still under construction, however it doesn't have registration yet, but if you are interested in seeing the website then you can login with demo/demo"** meaning that this was our way in through the first entry point. Taking the provided link and pasting it on the browser we are greeted by a webpage as shown below.

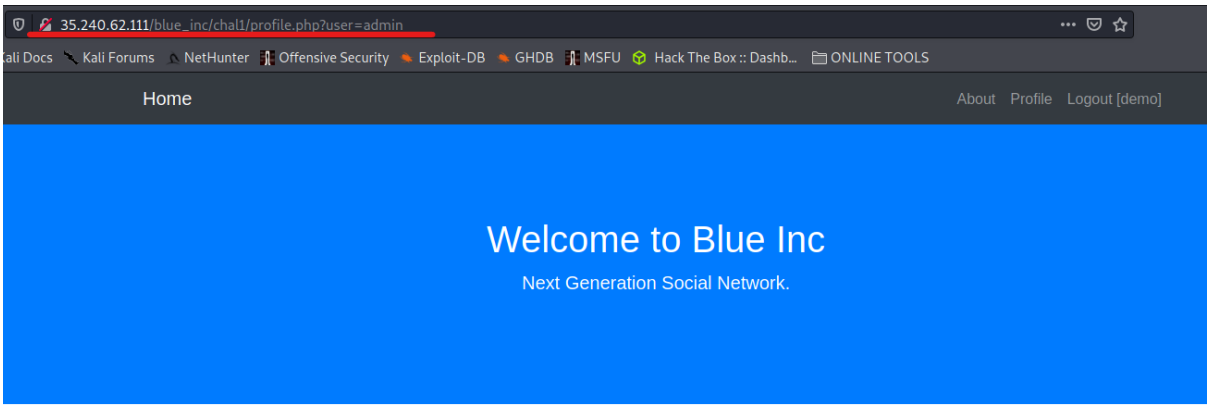


We clicked on the login button and entered our given credentials demo:demo and we logged in as demo. First thing that was visible to me after login in was the URL, I noticed that the link points to a page called profile.php which takes a parameter named user with value demo.



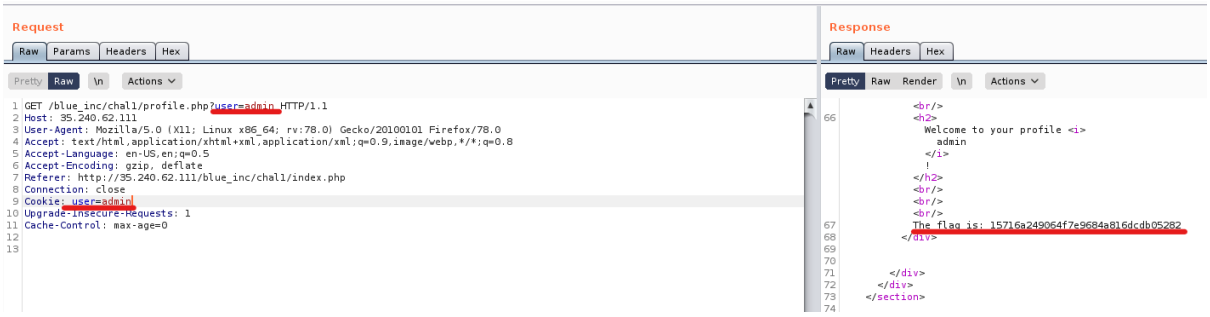


The next step was for me to try and modify the value from demo to admin, and after executing it I was given an access denied as shown below.



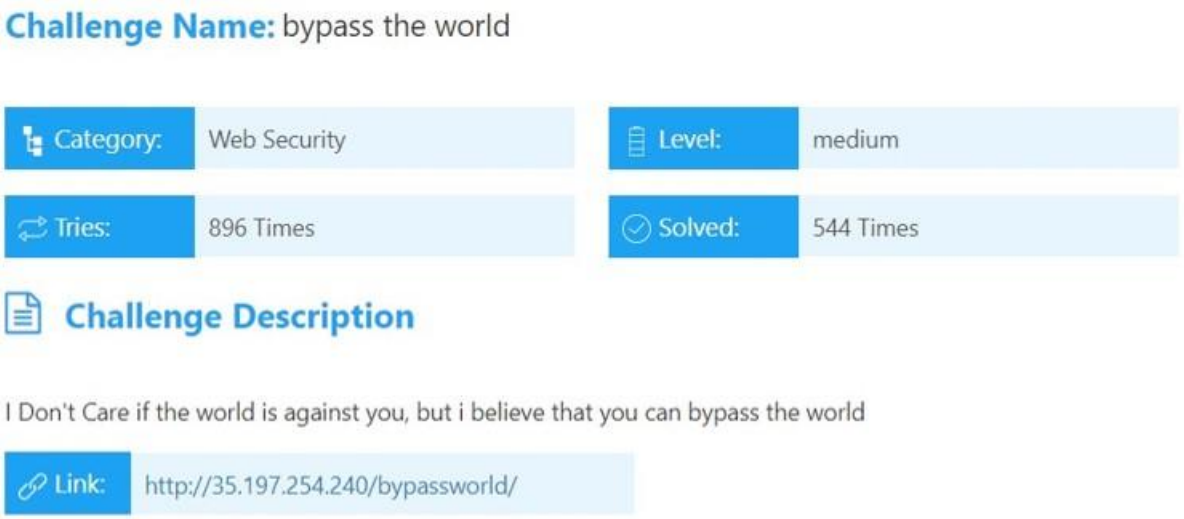
Next, I opened up burp suite and intercepted the request, and understood why initially my request was denied it was because I had changed only the URL value from demo to admin but there was a cookie which had the same parameter user and value demo which were supposed to be changed.

After making proper modification on the URL and cookie, when I sent the request to the server, I finally got the flag as shown below.



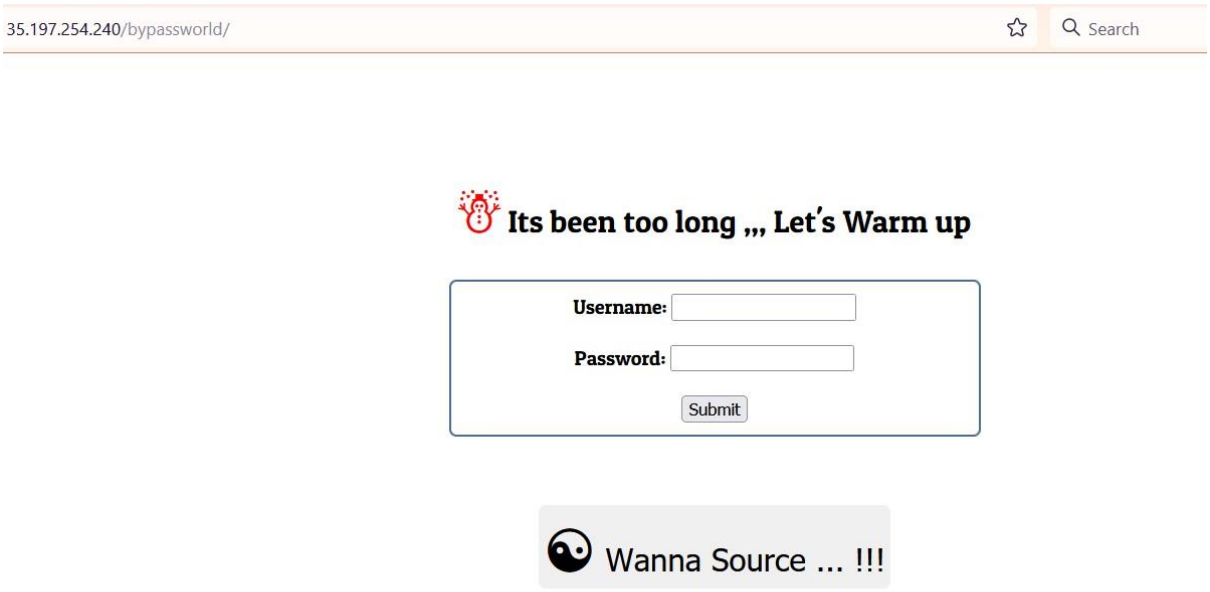
In this challenge we were exploiting a vulnerability in the access control and cookie

7. Bypass world



The next challenge was called bypass the world where its description says **that "I don't care if the world is against you, but I believe that you can bypass the world"** as the challenge says you can bypass the world, key word here is bypass meaning that the challenge may involve some sort of filtering mechanism bypass.

Looking at the provided link we are greeted by a login page. Tried login in with default credentials (admin:admin) but it didn't work.



Then I tried checking out the hit button which was called wanna source and we se a piece of code as shown below

according to the above piece of code the user authentication is validated through an SQL query. In the code we can see the string preg_replace. preg_replace searches for a string provided by the Source that matches the

regex Pattern and replaces it with the specified Replacement. The other part of the code shows the following pattern

/ is a delimiter that denotes the start and the end of the pattern

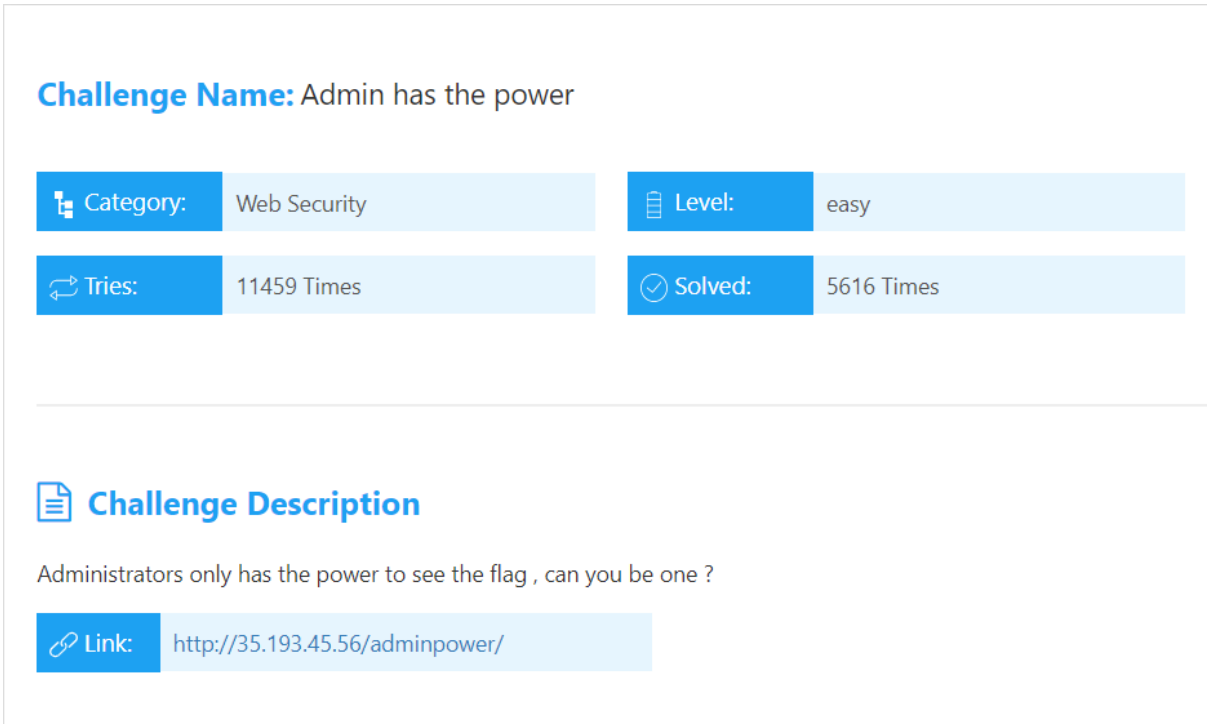
\ is used to escape the single quote and match ‘ literally

This pattern will simply remove any single quote , single quotes were going to be used to terminate that other single quote from the SQL query for both name & password so we can inject our query. since the query is a string, we can include a backslash (\) which is going to escape the single quote following the $name variable so the query would be something like

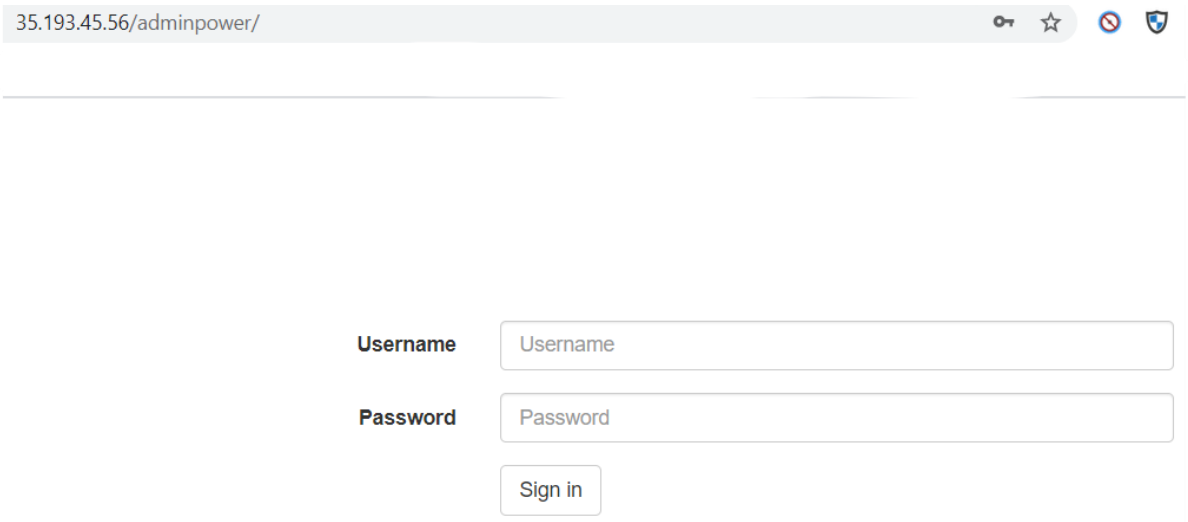
This payload could also be use **OR 1=1 -- -**

After executing the above payload which was going to bypass the filtering mechanism in place, and we got the flag as shown in the below image.

8. Admin has power



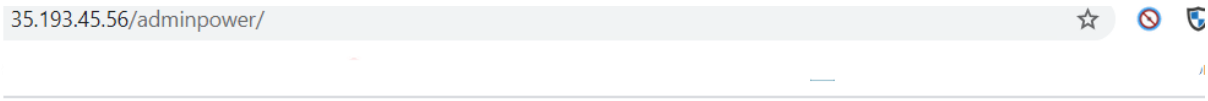
This challenge is called admin has power, which has the description of "**administrators only has the power to see the flag can you be one?**" looking at the provided link we see it a login page.

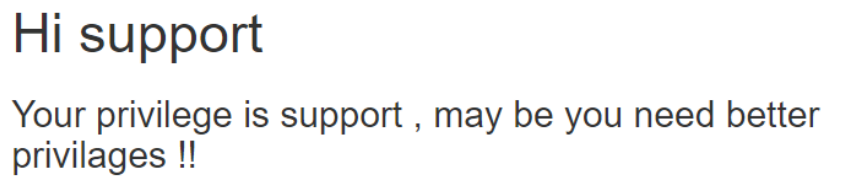


Looking at robots.txt and there was nothing useful there, next I went and inspected the source code and found credentials to login in the website page, sometime coders tend to insert hard coded comments so as to remember what part of the code does what and this practice can come with its consequence cause someone can forget to remove credentials in the hard coded comments as shown below.

```
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- The above 3 meta tags *must* come first in the head; any other head content must come *after* these tags -->
  <title>Admin Panel</title>

  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1d(

  <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
  <!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
  <![endif]-->
  <!-- TODO: remove this line ,  for maintenance purpose use this info (user:support password:x34245323)-->
</head>
<body>
```

Login in the page we see a message that says **"Hi support, your privilege is support, maybe you need better privileges"** this basically means that we need to gain access to the admin page.





Intercepting the request, we see that in the cookie there is a role which is set to support.

```
POST /adminpower/ HTTP/1.1
Host: 35.193.45.56
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0)
Gecko/20100101 Firefox/84.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Origin: http://35.193.45.56
Connection: close
Referer: http://35.193.45.56/adminpower/
Cookie: role=support; PHPSESSID=uq71pu04qd98uhnt4flff0jfa0
Upgrade-Insecure-Requests: 1

username=support&password=x34245323
```

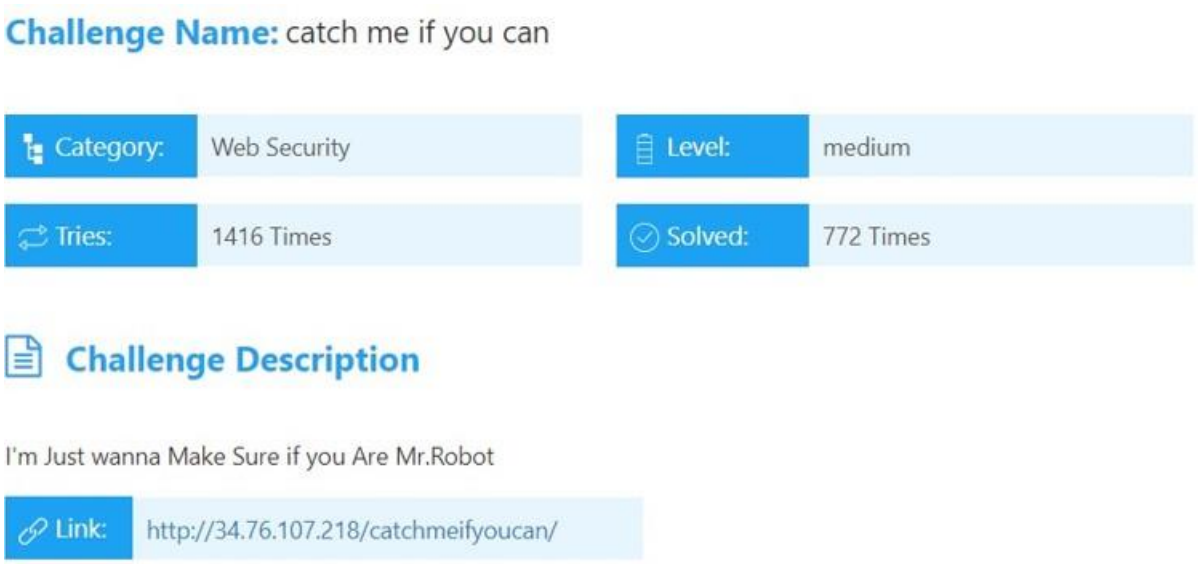
The next thing is to change the role from support to admin and then send the request back to the server

```
POST /adminpower/ HTTP/1.1
Host: 35.193.45.56
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Origin: http://35.193.45.56
Connection: close
Referer: http://35.193.45.56/adminpower/
Cookie: role=admin; PHPSESSID=uq71pu04qd98uhnt4flff0jfa0
Upgrade-Insecure-Requests: 1

username=support&password=x34245323
```

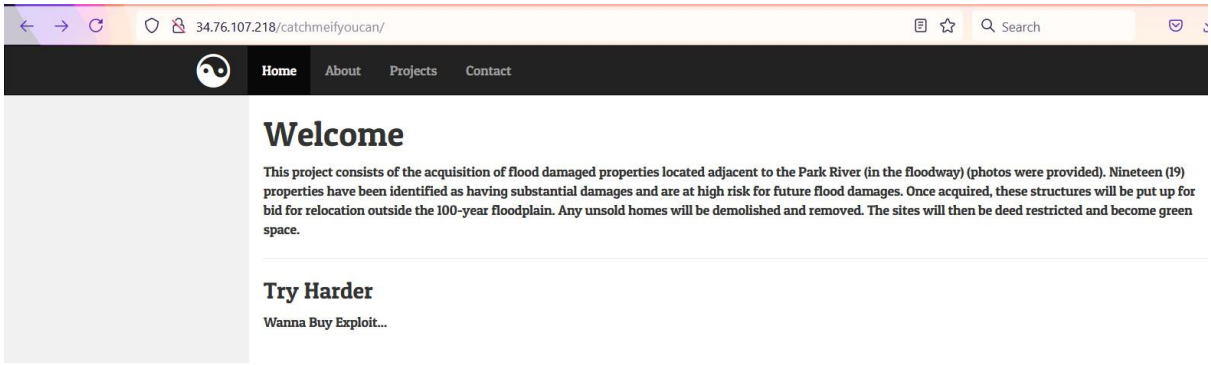
After sending the request to the server and we inspect the result and get the flag as shown below

```
</head>
<body>
  <div class="container" style="padding-top    :150px;">
    <div class="row">
      <div class="col-sm-6 col-sm-offset-3">
        <h1>
          Hi admin
        </h1>
        <h3>
          Admin Secret flag : hiadminyouhavethepower
        </h3>
      </div>
```

9. Catch me if you can

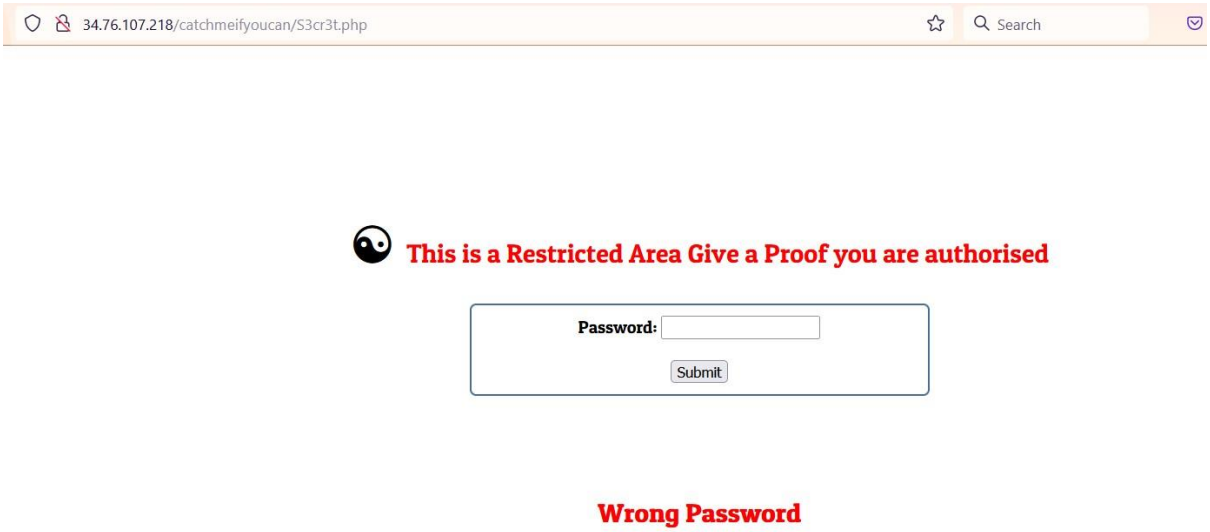


The next challenge was called catch me if you can with the description of **"I'm just wanna make sure if you are not Mr. robot"**. we take the URL provided and run it, and we see that we have a website.

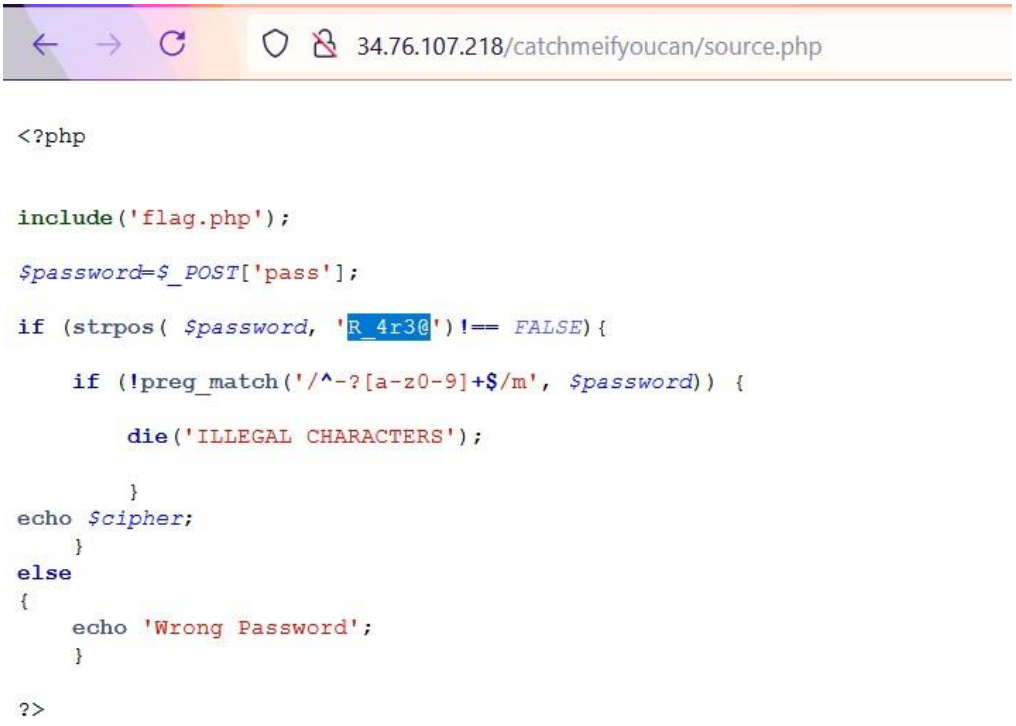


I started by navigating round the website and there was nothing interesting, then I went and examined the source code and there was nothing interesting there too and finally trying robots.txt I found there were two files as shown below.

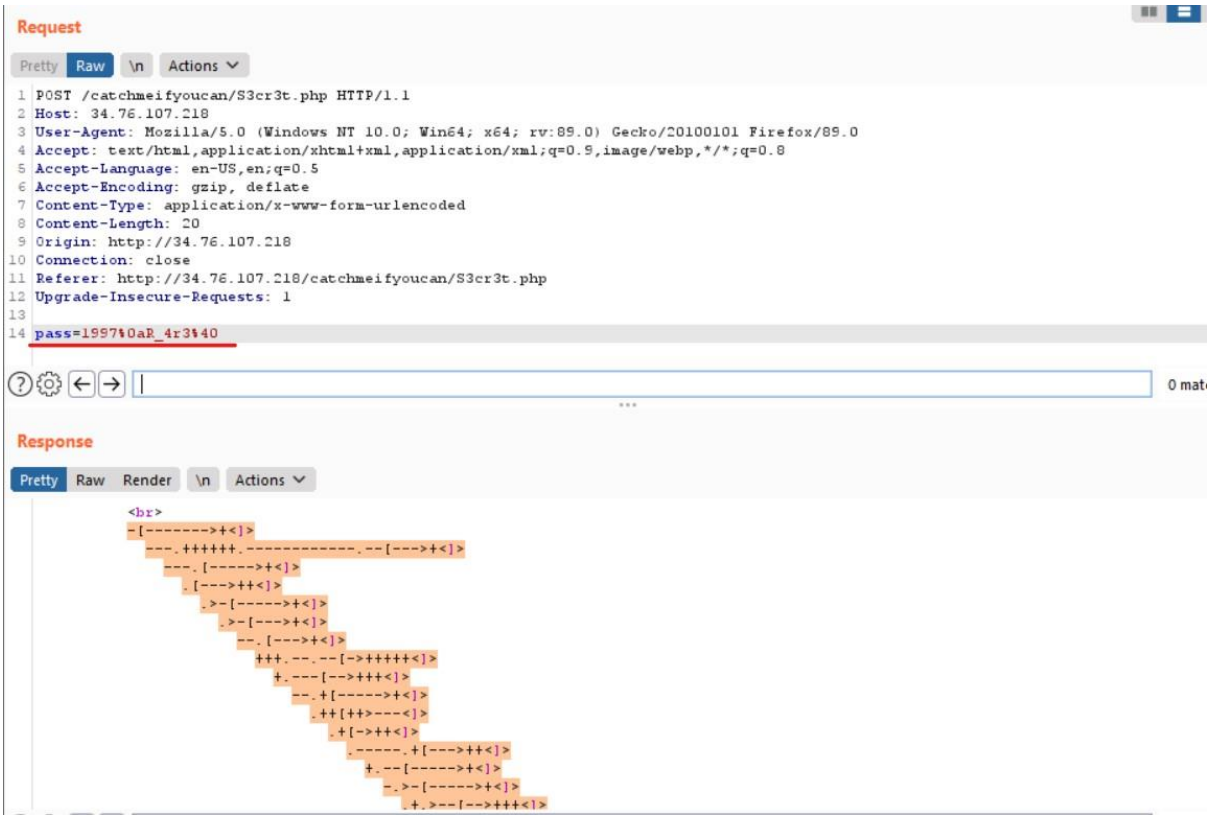
Checking out the path **/S3cr3t.php** and find that there is a login page asking for a password.



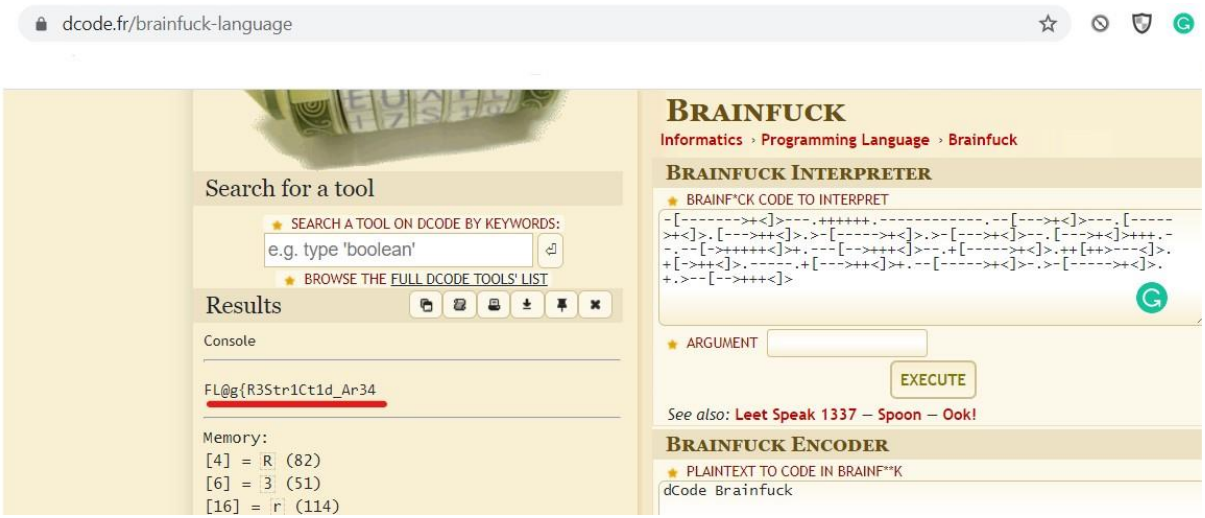
Going to the second path **/source.php** we see that there is a code hinting to the login page. Based on the code the password must end with **R_4r3@** and must contain numbers letters from a to z and numbers from 0 to 9



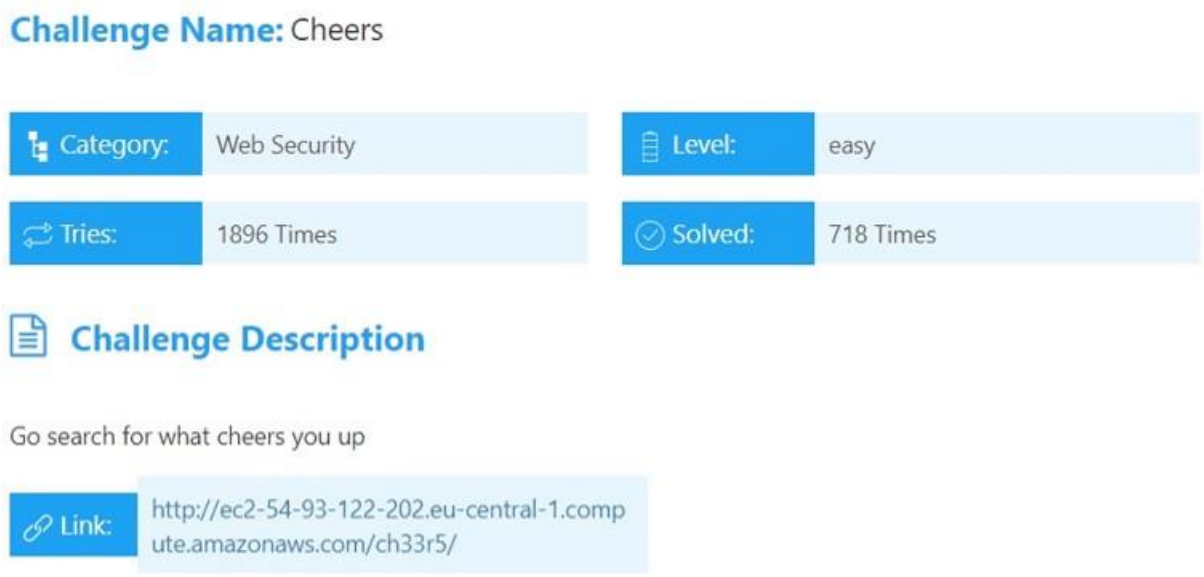
So, for this we need a function to bypass (break) the preg_match function which for our case we can use the value **%0a** (%0a is a URL encoding of a new line) and based on the rule to bypass the filter out password would be **1997%0aR_4r3%40 or 1997%0aR_4r3@**



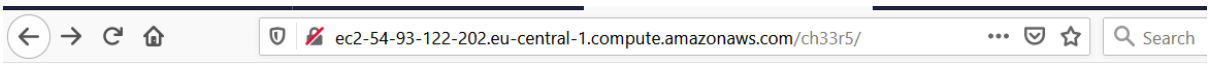
And we get a response of an obfuscated code which is called **brain fuck** for us to decode in order to get the flag. Sending the codes in an online decoder we get our flag as shown in the image below



10. Cheers



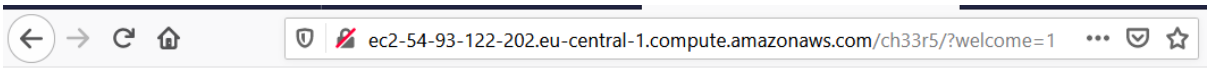
The next challenge was called cheers which descriptions says **"go search for what cheers you up"** funny description this one has but I opened the link and a webpage was on.



Ops!! Can You Fix This Error For Me Please ??!

Notice: Undefined index: welcome in /var/www/html/ch33r5/index.php on line 14

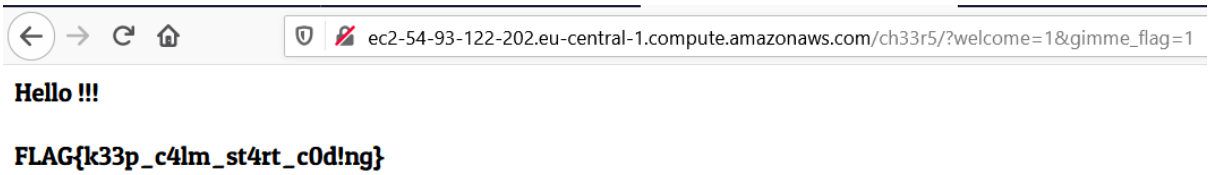
And we see that there is a php error on page, now let us try to fix it. From the error message from the page, it says there is an undefined index called welcome. So, I took the parameter welcome and inserted it in the URL like **ip/ch33r5/?welcome** and got another php error page.



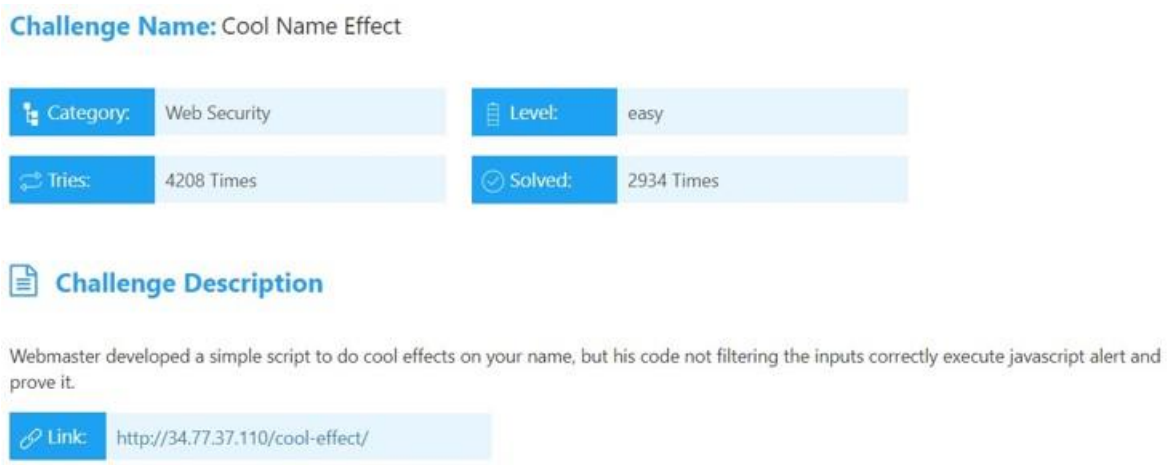
Hello !!!

Notice: Undefined index: gimme_flag in /var/www/html/ch33r5/index.php on line 19

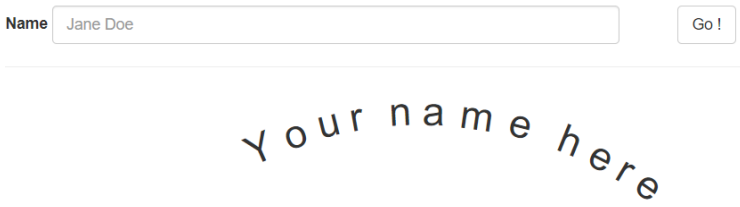
Next I sent both the first unified index and the second one which looked like **ip/ch33r5/?welcome&gimme_flag** and got the flag.



11. Cool name effect



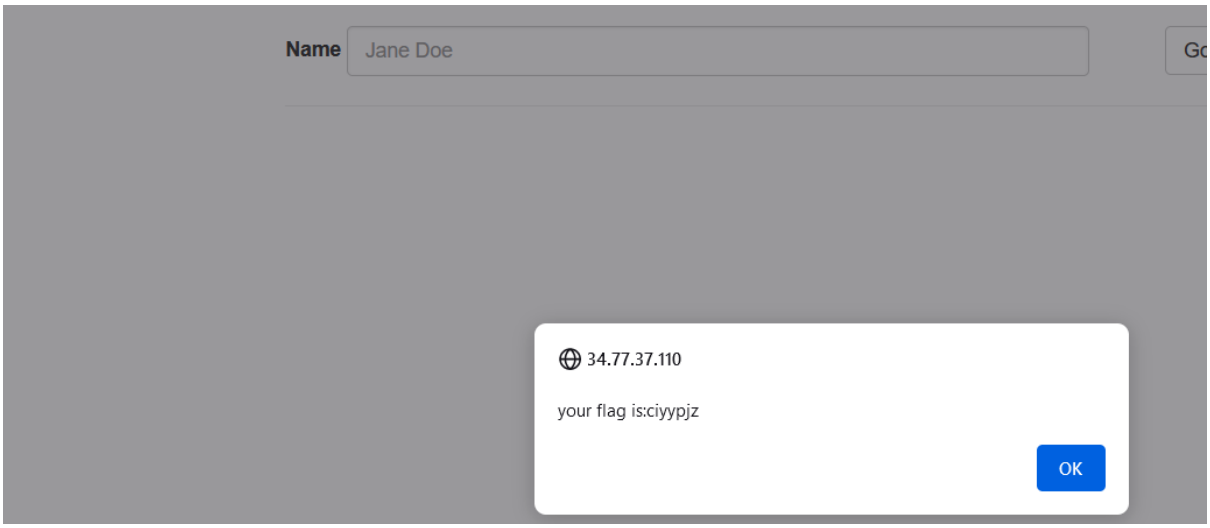
The next challenge was called cool name effect with the description of "**webmaster developed as simple script to do cool effects on your name, but his code not filtering the inputs correctly execute java script alert and prove it**" this challenge basically wants us to perform a cross site script (XSS) on the page. Opening the page we see a website where you can type anything.



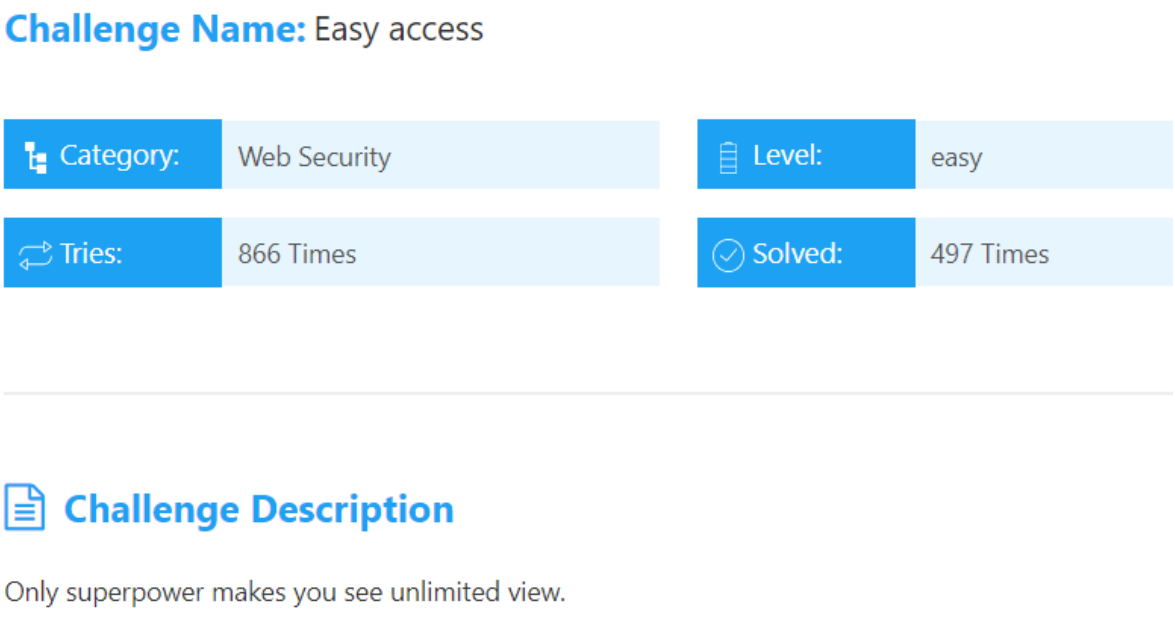
I started by typing the word test and see what will happen and see that it displayed the string on the page as shown on the below image.

Next, I had sent a XSS payload and see how the page reacts and we can see that the word script is being filtered and being replaced by the word forbidden, this means that there is some sort of filtering mechanism at the backend that does this.

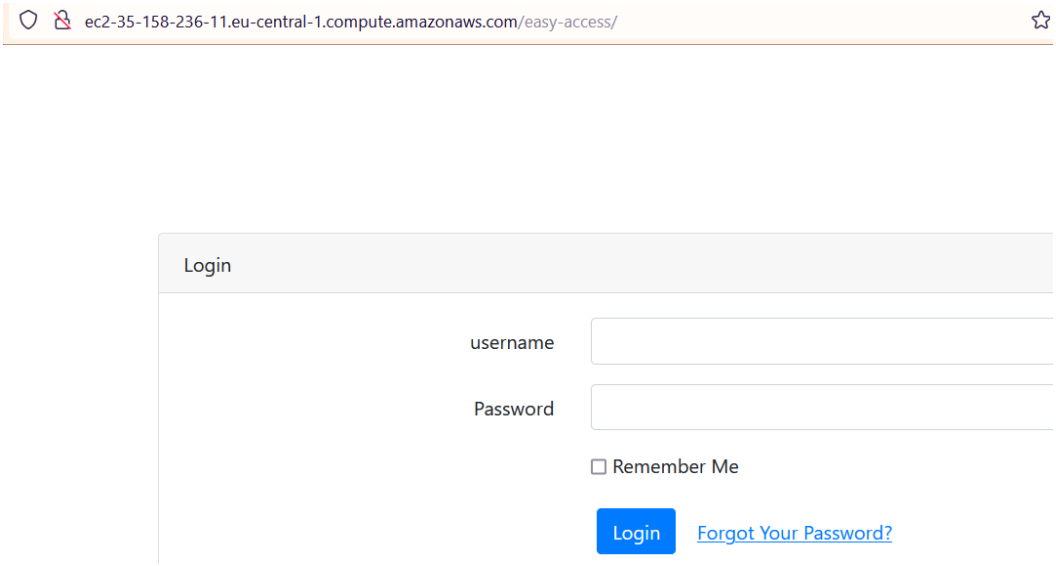
For this we can make use of HTML tags which are case insensitive, this means that we can make the script tag letters with a mixture of lower and upper-case letters which is going to be interpreted no different than a lower-case script tag. The script would be as follows <ScRiPt>alert(1)</ScRiPt> and by executing this it bypasses the current filter and pop up an alert which give us our flag as shown in the image below.



12. Easy Access



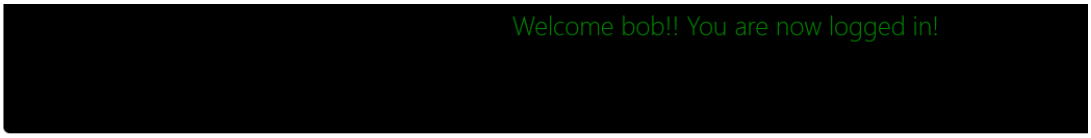
The next challenge was called Easy Access which had a description of **"Only superpower makes you see unlimited view"** from the description they are referring to the admin page. Went to the link they had provided, and a login page appeared.



Tried to check robots.txt but it had nothing useful from there, next I went to examine the source codes and found hard coded credentials on the web site.

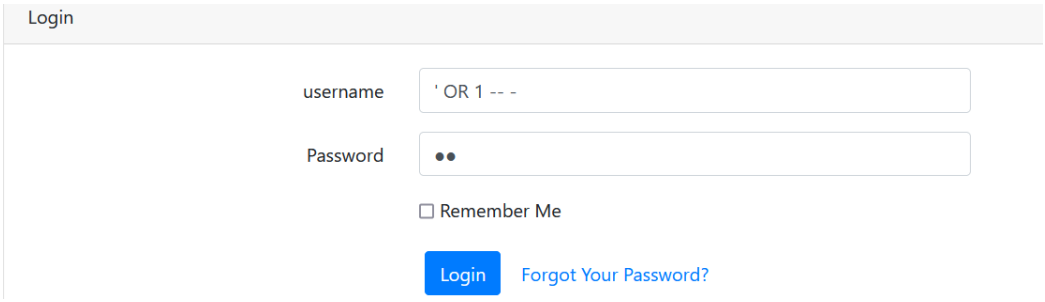
```
    <!------ Include the above in your HEAD tag ---------->
    <!-- Fonts -->
    <link rel="dns-prefetch" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Raleway:300,400,600" rel="stylesheet" type="text/css">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">

    <title>easy_access</title>
</head>
<!---user:bob,pass:password-->
<body style="height:auto;margin:0 auto;padding:0 auto ;">
```

Logging in the page and see the page says only the admin can see the flag, intercepted the traffic using burp and there was nothing that I could tamper with to try and exploit the page

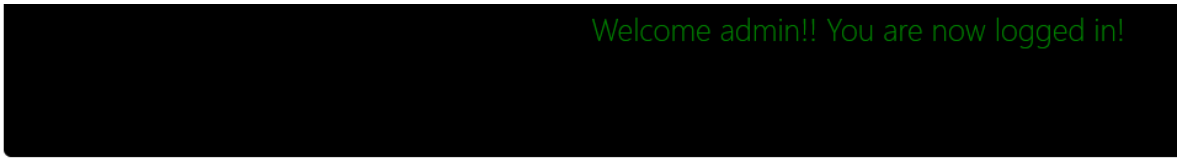


Only admin can see the flag

Went back to the page and tried to use SQL Injection on the page to try and exploit it, I inserted the payload on the username and anything on the password cause it won't affect anything due to the -- - which is going to ignore everything else after.

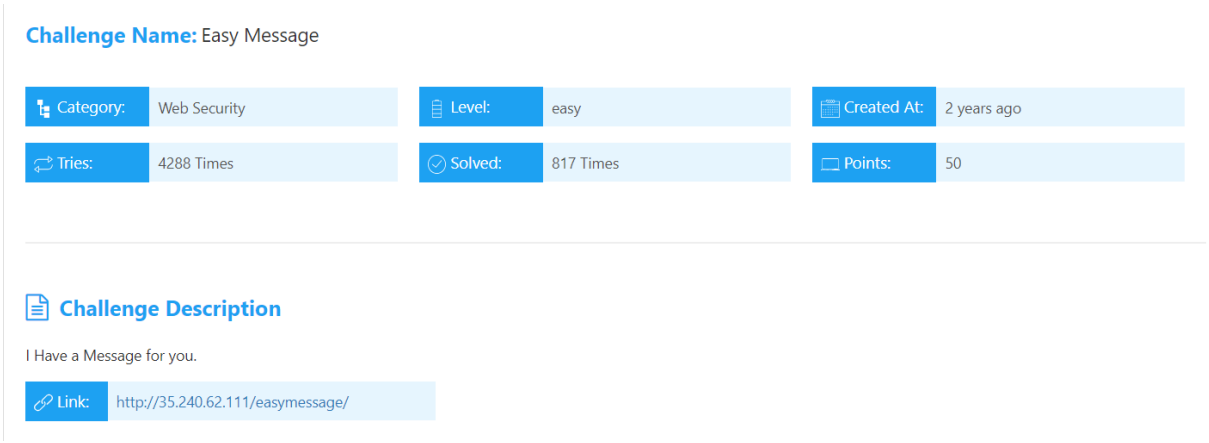


After executing the payload we get a welcome as admin and see the flag

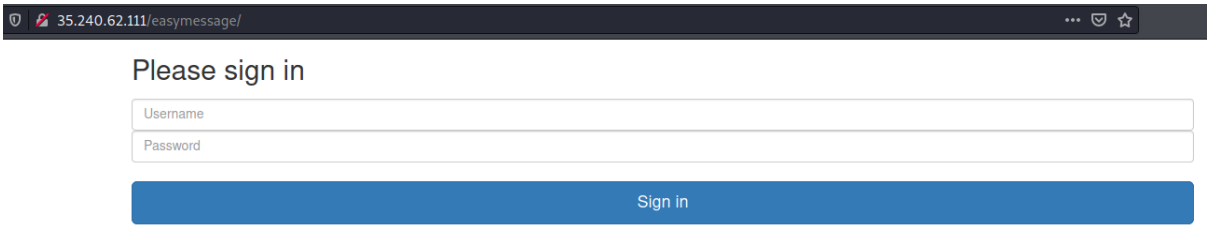


flag{!njection_3v3ry_wh3r3}

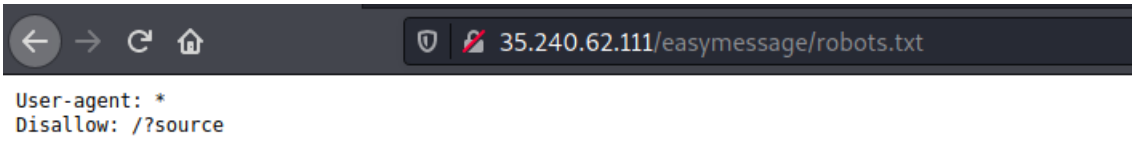
13. Easy Message



The next challenge was called Easy Message which had a description of **"I have a message for you"** going to the link that was provided I found it was a login page.



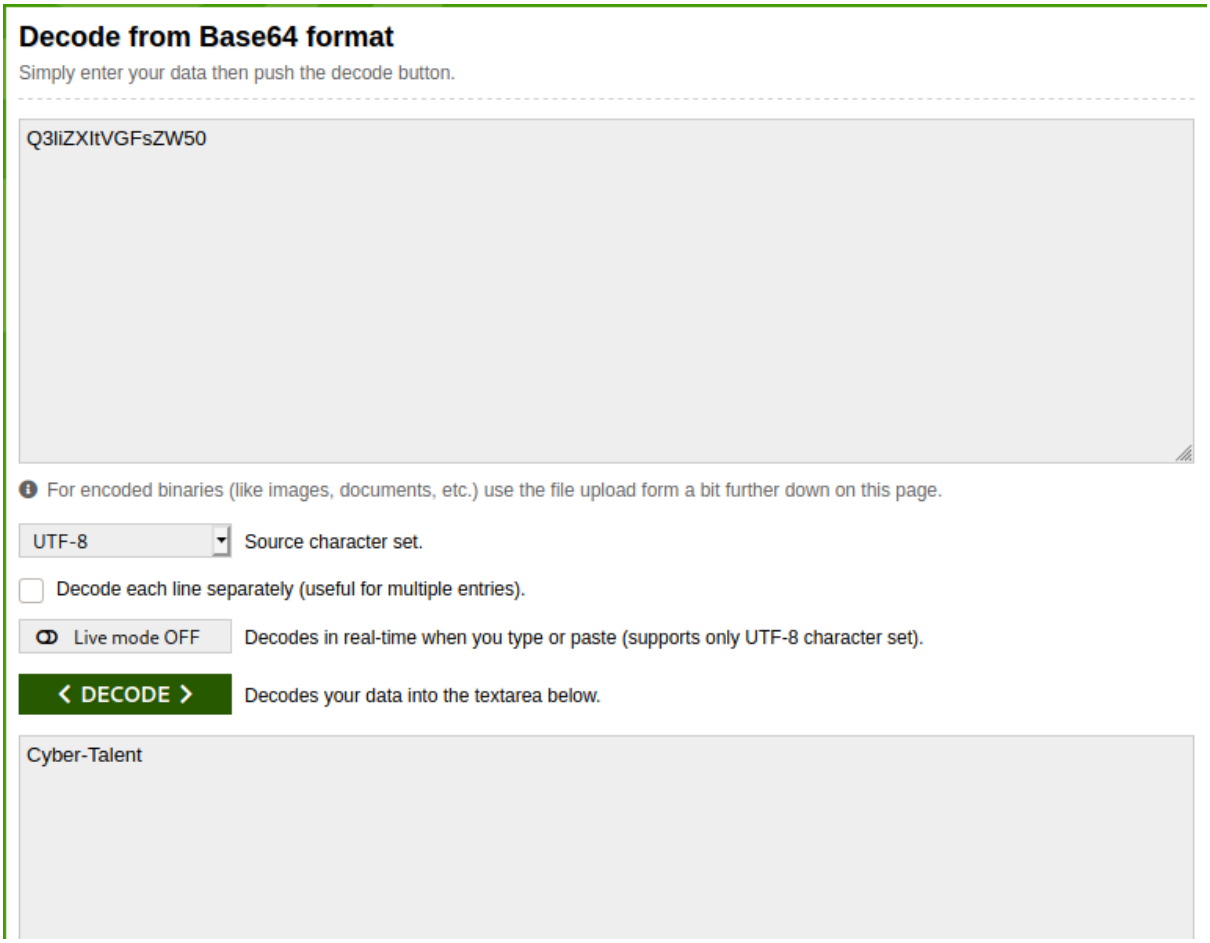
Going through the source page there was nothing interesting, inspected element and there was nothing interesting there too, I decided to go and do a directory brute force and found there was a robots.txt available and which I viewed the path and found there was an interesting file path called source as shown below.



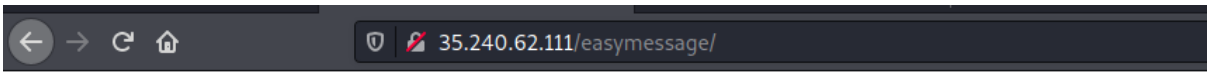
Though the folder says disallowed but what may be disallowed doesn't mean its actually not accessible, so I went, and copy pasted the path to the URL and found the source code of the login page.

The code basically says that the username and password are the same which are basically encoded in base64 as shown in the image below.

Next, I took the base64 value from the code and decoded it got the username and password to be Cyber-Talent



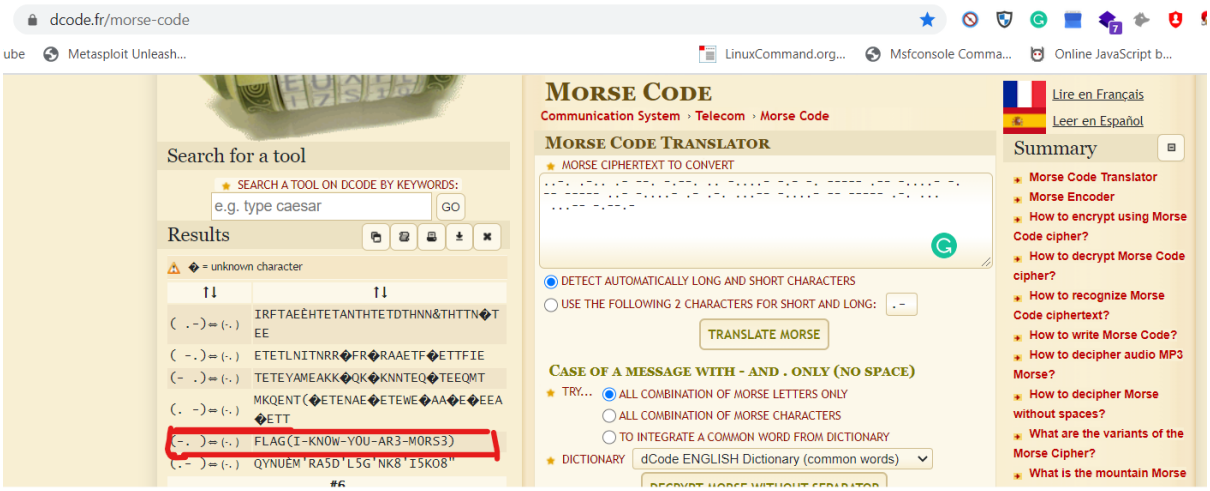
I used the credentials to login the website and once I was in I had found that there was a Moores code left on the page



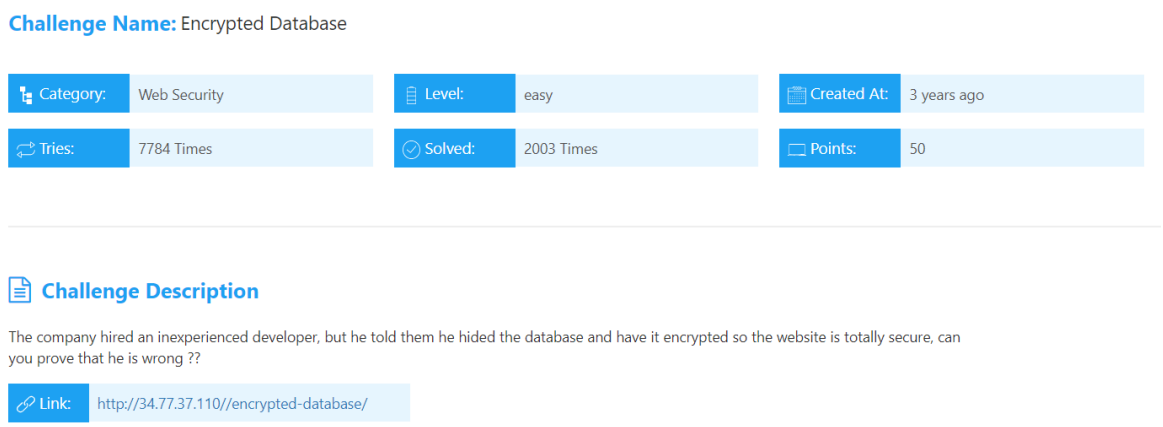
I Have a Message For You

..-. .-.. .- --. -.--. .. -....- -.- -. ----- .-- -....- -.-- ----- ..- -....- .- .-. ...-- -....- -- ----- .-. ... ...-- -.--.-

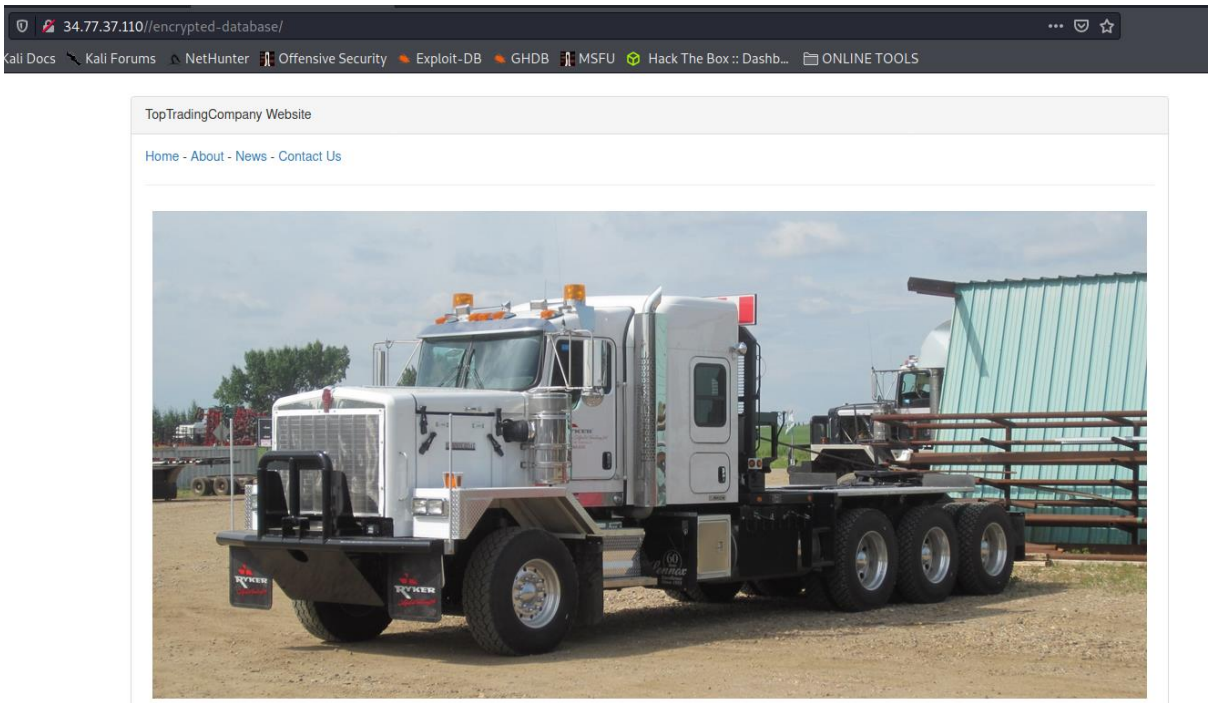
I took the mores code and went to an online decoder and decoded the Moores message and finally I obtained the flag.



14. Encrypted Database

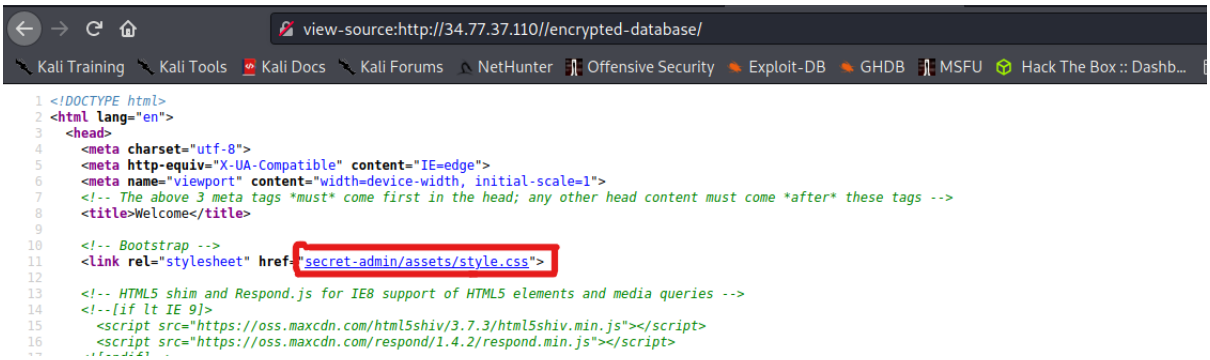


The next challenge was called Encrypted Database which had a description of "**the company hired an inexperienced developer, but he told them the database, and have it encrypted so the website is totally secure, can you prove that wrong**" going to the link that was provided I found it was a website with an image of a track.

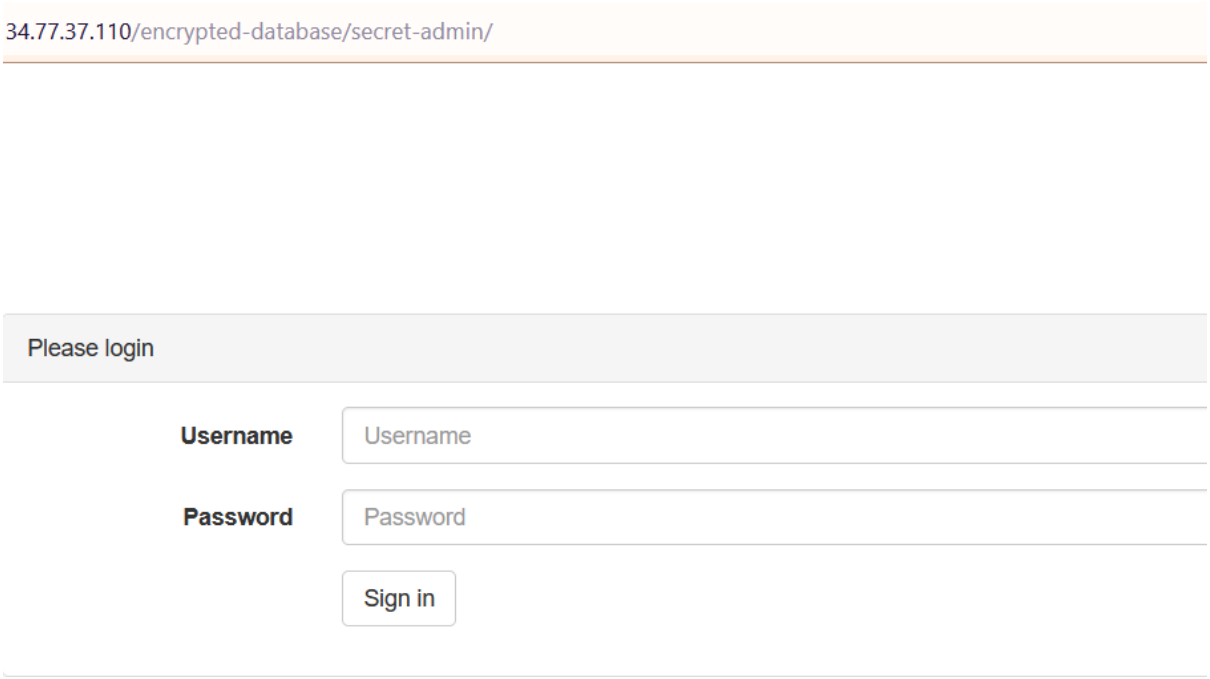


Going through the page and there was nothing interesting nothing interesting there so I had to view the source page.

View the source page I had seen a link which was of potential interest, it had a path called secret-admin which had some css codes.

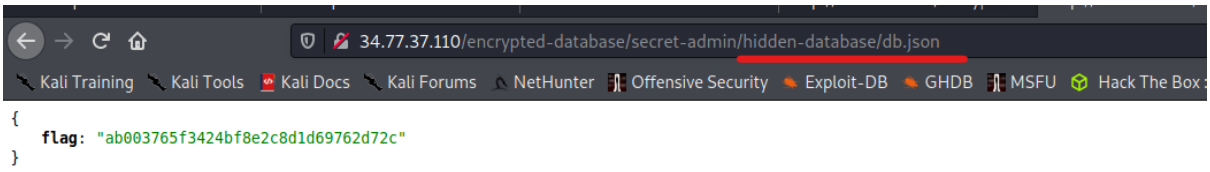


Going to the secret-admin path and we found ourselves with a login page.

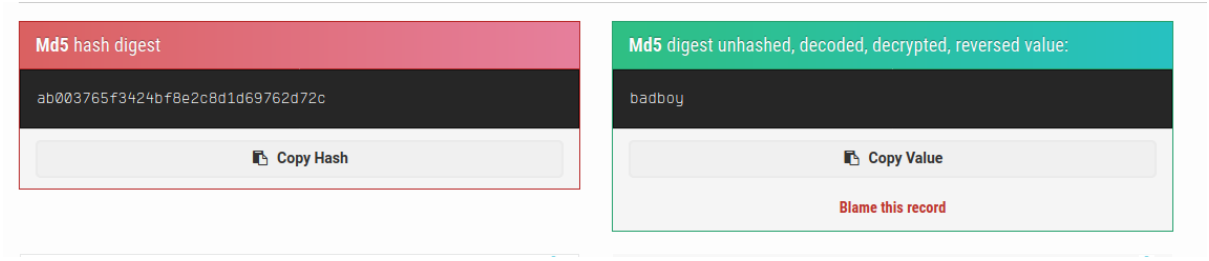


Viewing the source code, I found another path which was called hidden-database/db.json and this was the database that the inexperienced developer said it was secured.

Going to the path of the database and we found the flag, inserting it to the hash identifier in kali linus and it said it was an MD5 cryptographic algorithm.



Decrypting the MD5 and we got our flag as shown in the image below.



15. Got control

Challenge Name: Got Controls

Category:	Web Security	Level:	easy
Tries:	406 Times	Solved:	349 Times

Challenge Description

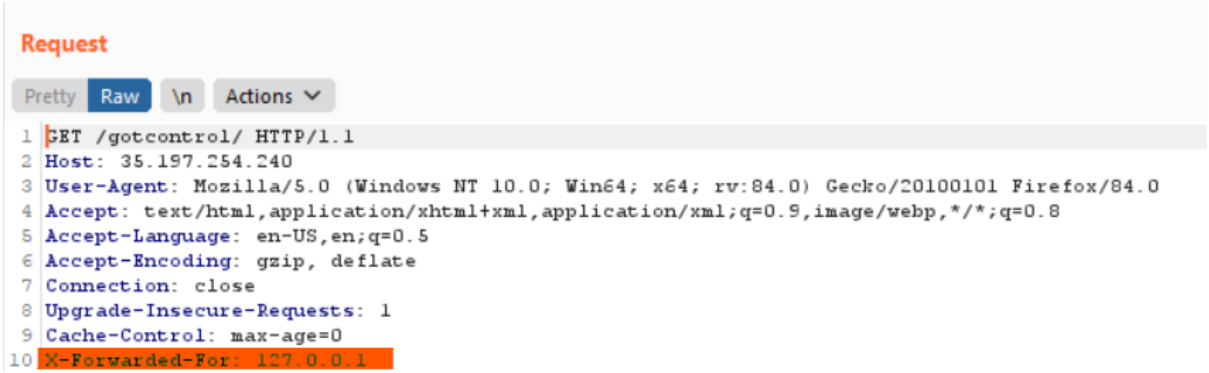
We believe we made a good job protecting our infrastructure, can you bypass our controls.

Link: http://35.197.254.240/gotcontrol/

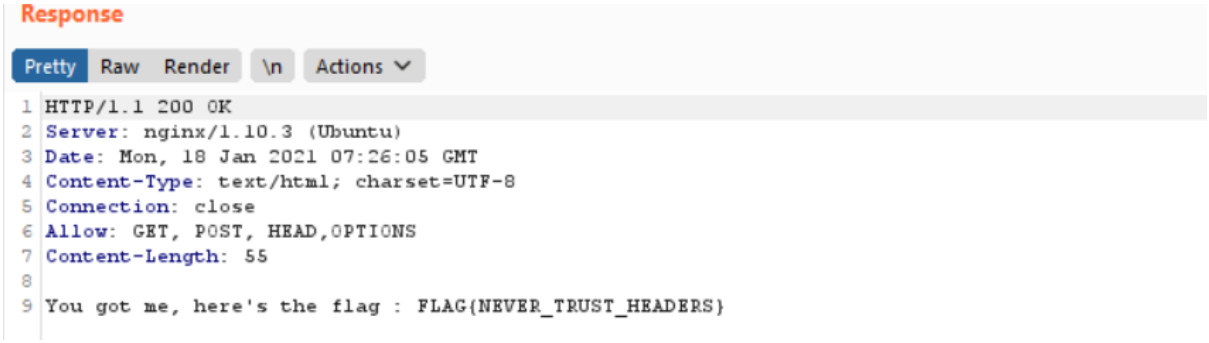
The next challenge was called Got Control which had a description of "**we believe we made a good job protecting our infrastructure, can you bypass our controls**" going to the link that was provided I found it was a page which stated that "**sorry your ip is not allowed, this server is only accessible from local machine or local LAN** as shown in the image below

Looking at the source code of the page and there was nothing there, looking at the message it says its only accessible from the local machine which means 127.0.0.1 or localhost. Basically, we are dealing with headers Next step we need to use user IP where I googled "**http header to get user ip**" and got information about X-FORWARDED-FOR header which is basically a de-facto standard **header** for identifying the originating **IP** address of a **client** connecting to a web server through an **HTTP** proxy or a load balancer

To use this header all you have to do is on burpsuite you type X-Forwarded-For: 127.0.0.1 in the request header as shown below



And then you send the request, and we get the flag



16. Hashable

Challenge Name: Hashable

Category:	Web Security	Level:	medium
Tries:	1562 Times	Solved:	1047 Times

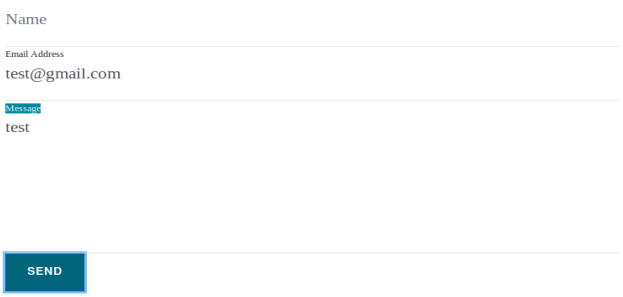
Challenge Description

A famous enterprise blog was hacked, can you figure out how it was hacked?

Link: http://ec2-35-158-236-11.eu-central-1.compute.amazonaws.com/hashable/

The next challenge was called Hashable which had a description of "**A famous enterprise blog was hacked; can you figure out how it was hacked?**" going to the link that was provided I found it was a website

Going through the tabs that were available all had nothing of interest except contact. I found a form which was the main point of entry into the website.



Sending the request to burpsuite, I initially inserted single quotes and later double quotes to see what response I would get and with the double quote the page returned with an error as shown on the below image.

Error in: /var/www/html/Hashable/contact.php(27) : eval()'d code
Message: syntax error, unexpected '"', expecting ',' or ')'

You can notice that the page makes use of the eval() function which pose a threat to a website, from this we know we can test for code injection. Code injection is the exploitation of a computer bug that is caused by processing invalid data. Now we can inject a code that will show who the user is in the server.

Request

Raw | Params | Headers | Hex

```
POST /hashable/contact.php HTTP/1.1
Host: ec2-35-158-236-11.eu-central-1.compute.amazonaws.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://ec2-35-158-236-11.eu-central-1.compute.amazonaws.com/hashable/contact.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Upgrade-Insecure-Requests: 1

name=&email=test%40gmail.com&message=test${system("whoami")}
```

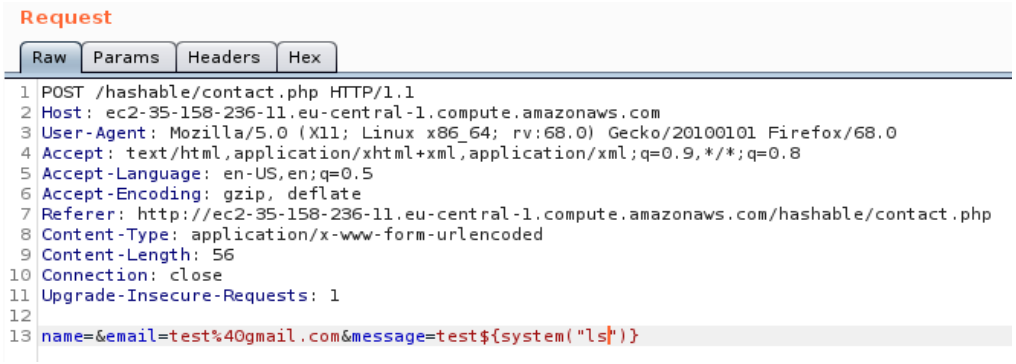
And we got a respond as shown in the below image meaning that the exploit will work.

Response

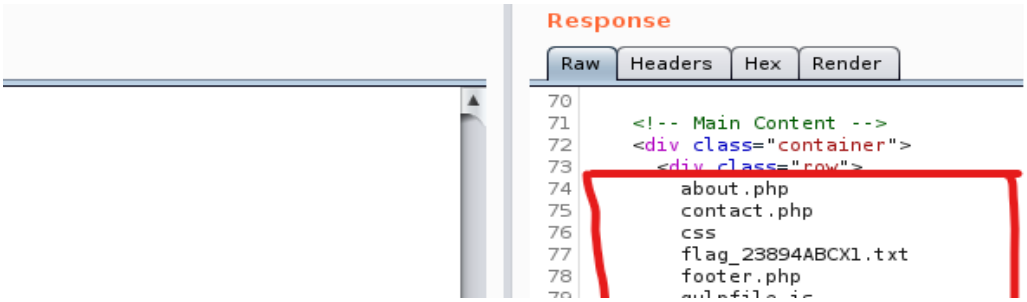
Raw | Headers | Hex | Render

```
        </div>
      </div>
    </header>

    <!-- Main Content -->
    <div class="container">
      <div class="row">
        www-data
        Message successfully sent!        <div class="col-lg-8 col-md-10 mx-auto">
          <p>
            Want to get in touch? Fill out the form below to send me a message and I wil
```

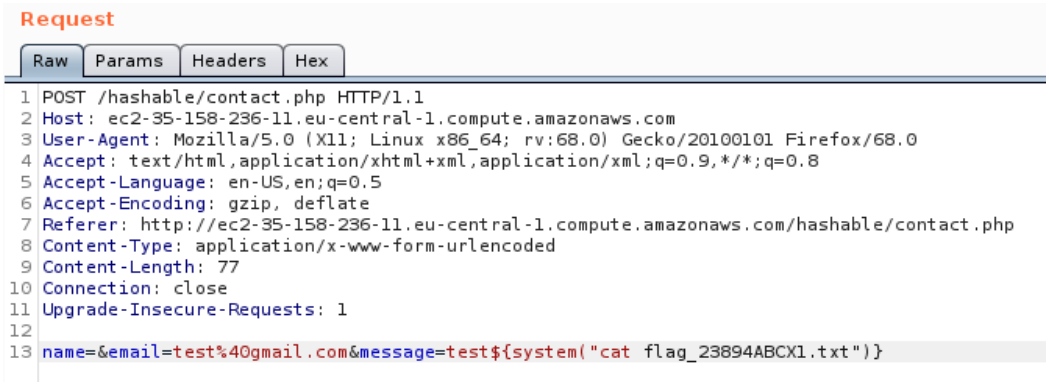
Next, we injected a code that list all files in the server directory as shown below



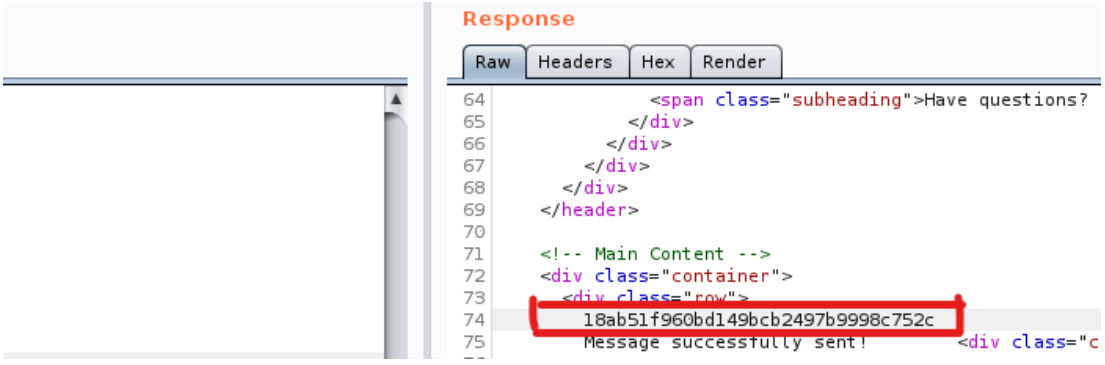
And we got the list of files as shown below and one of the files was the flag



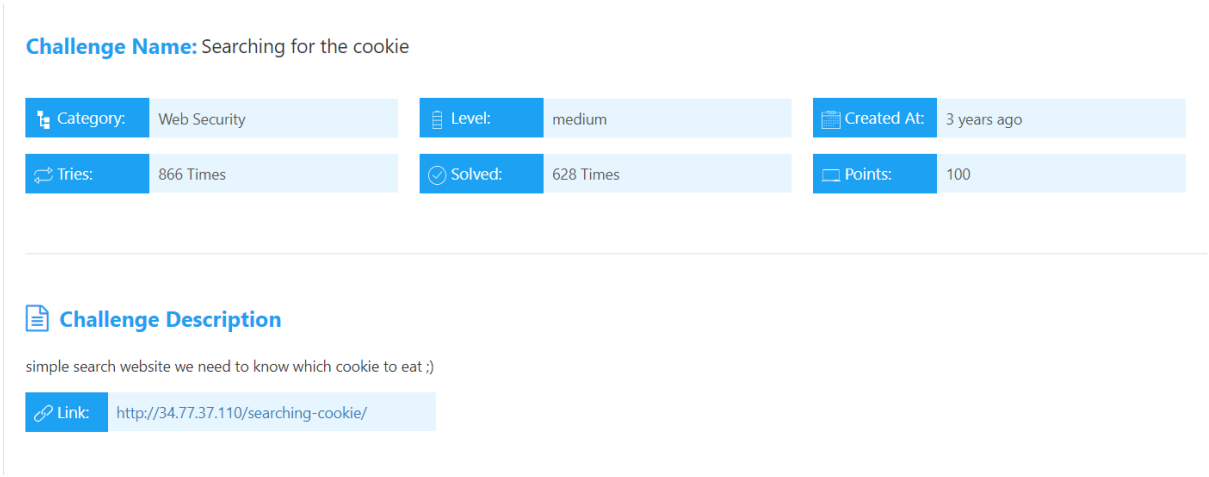
Then we had cat the flag using the code we had injected as shown in the image below



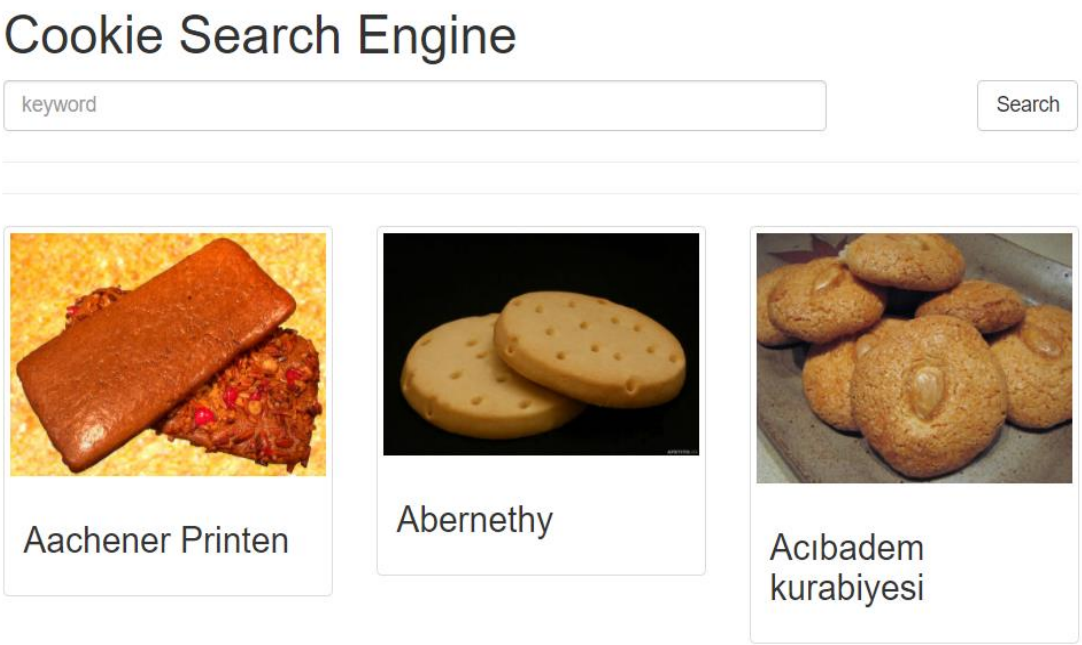
And we finally got the flag.



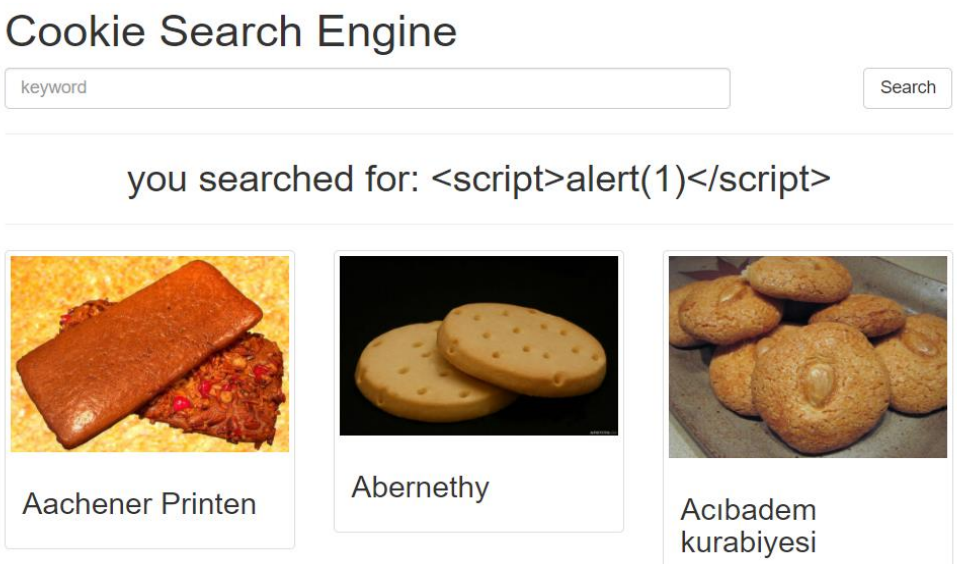
17. Search for the cookie



The next challenge was called Searching for the cookie which had a description of "**simple search website we need to know which cookie to eat :)**" going to the link that was provided I found it was a website



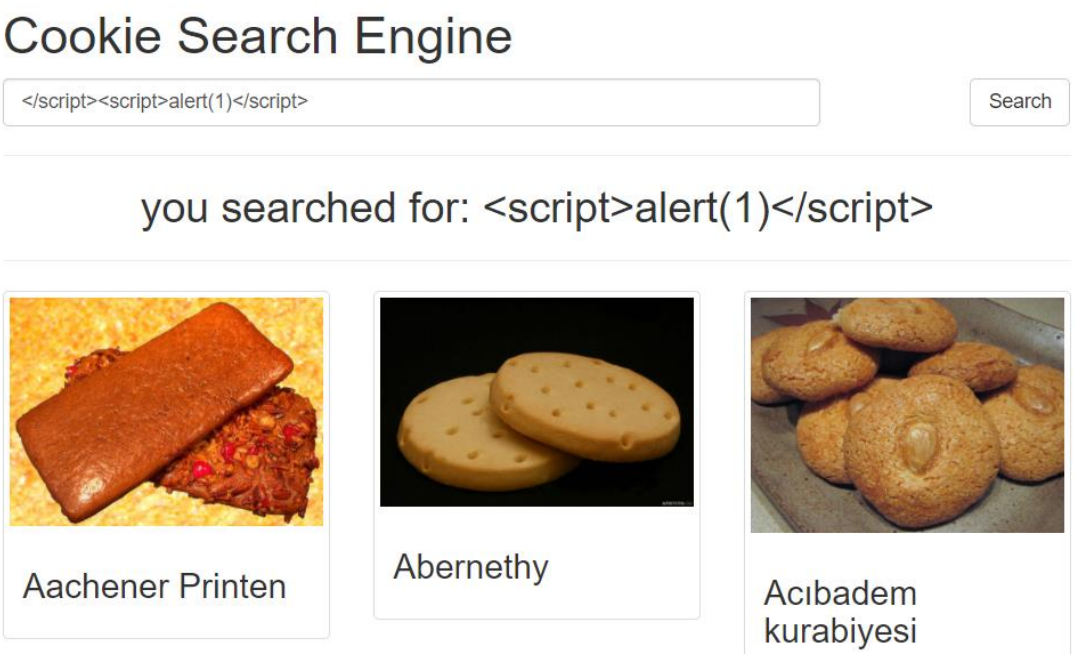
The website had one entry point which is the search tab, and it was where I was going to attack first. I executed a XSS payload on the page and the payload was not successful



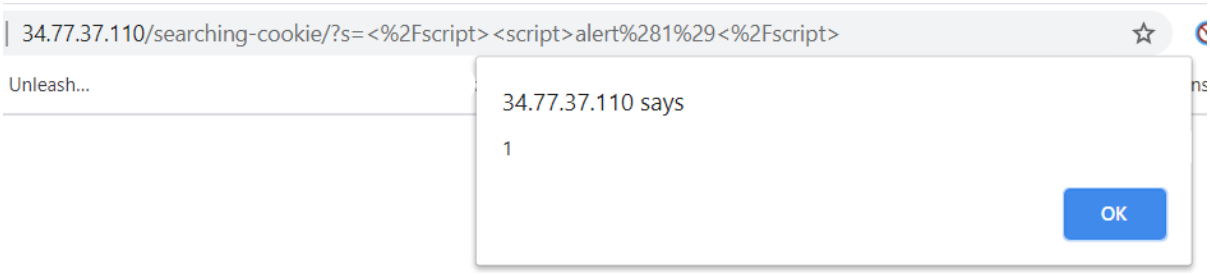
Looking at the source code I noticed that the search function is implemented using JavaScript and the malicious script we had send from the search bar was placed inside that script hence we conclude that it is a DOM Based XSS, I had to know how to inject your script so I can exploit the bug

```
<script>

eval(function(p,a,c,k,e,d){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toString(36))};if(!''.r

</script>
<script>var currentSearch = {'keyword':'you searched for: <script>alert(1)</script>'};</script>

</head>
<body>
    <div class="container" style="padding-top   :100px;">
        <div class="row">
            <div class="col-sm-8 col-sm-offset-2">
                <h1>Cookie Search Engine</h1>
                <form class="form-inline" method="get">
                  <div class="form-group" style="width:90%;">
                    <input style="width: 85%;" type="text" name="s" class="form-control" id="exampleInputName2" placeholder="keyword">
                  </div>
                  <button type="submit" class="btn btn-default">Search</button>
                </form>
                <hr />
                <h2 class="text-center">
                    you searched for: &lt;script&gt;alert(1)&lt;/script&gt;                </h1>
```

in order to pop up the cross-site script we needed to come up with a payload that would bypass the script and pop up the alert. I tried using end scripts with an opening script as shown below

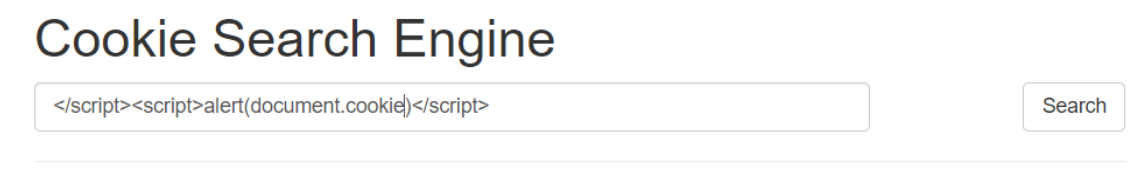


It eventually popped up an alert as shown in the image below

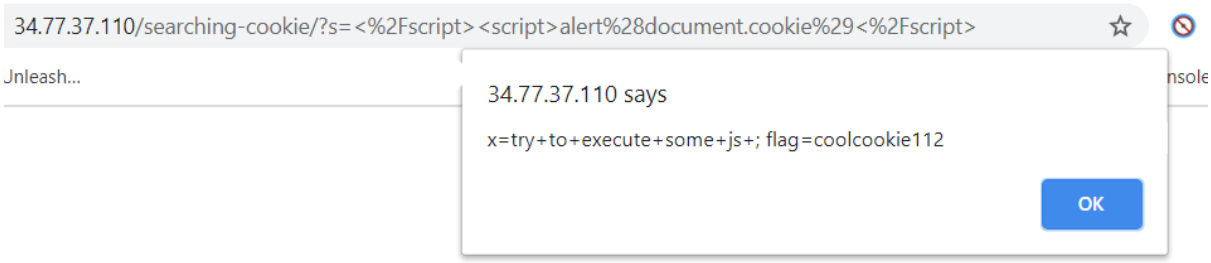


Since the challenge is asking to exploit the cookie then we are going to use the document.cookie payload

Executing the new payload as shown below



We managed to pop up the alert and we obtained the flag



18. Secret browser

Challenge Name: Secret Browser

Category:	Web Security	Level:	medium
Tries:	473 Times	Solved:	362 Times

Challenge Description

The company employees is using company special browser to view the website content.

Link: http://35.197.254.240/secret-browser/

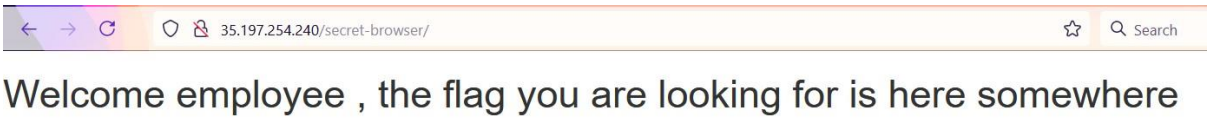
The next challenge was called Secret Browser which had a description of "**The company employees is using special browser to view the website content**" going to the link that was provided I found it was a website that stated that I am not using the company browser.

Inspecting the code and found the company name **PublicTradeCo**

I intercepted the request that was supposed to be send to the server and added the header User-Agent and company name PublicTradeCo as shown below

```
GET /secret-browser/ HTTP/1.1
Host: 35.197.254.240
User-Agent: PublicTradeCo
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/w
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

And the request was sent successfully and got the welcome employee message but unfortunately, we had to look for the flag since it was not on the home page of the website as shown below



After several hours of looking for the flag, I had found the flag as shown below

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Thu, 01 Jul 2021 09:01:40 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
x-flag: W3lcomeCOmpanyUs3R
Allow: GET, POST, HEAD,OPTIONS
Content-Length: 1408
```

19. Ugame

Challenge Name: uGame

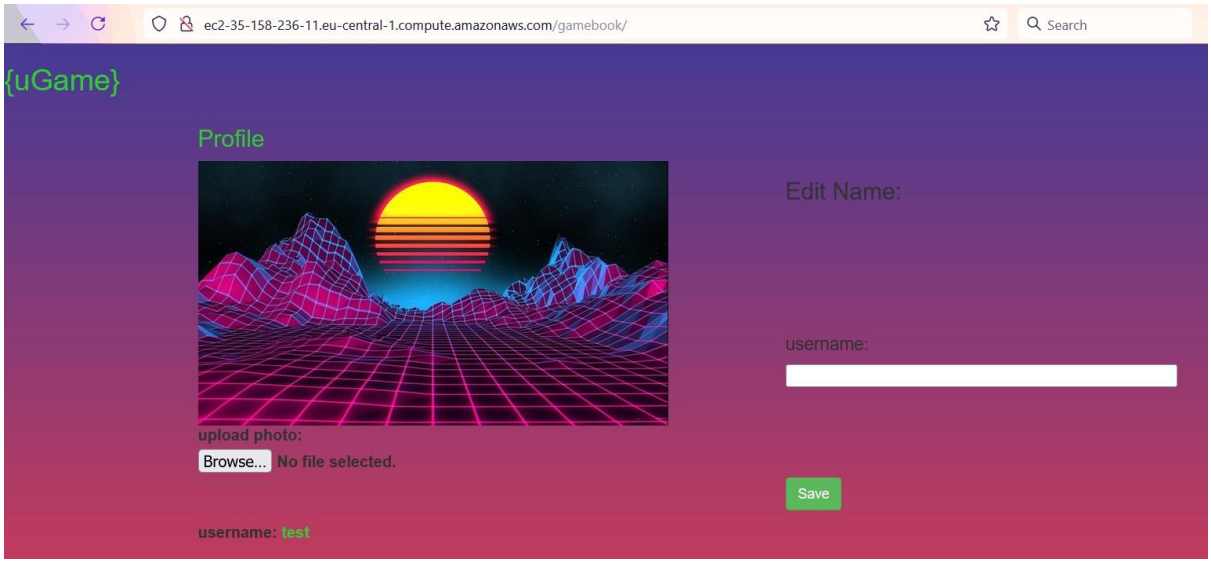
Category:	Web Security	Level:	easy
Tries:	358 Times	Solved:	262 Times

Challenge Description

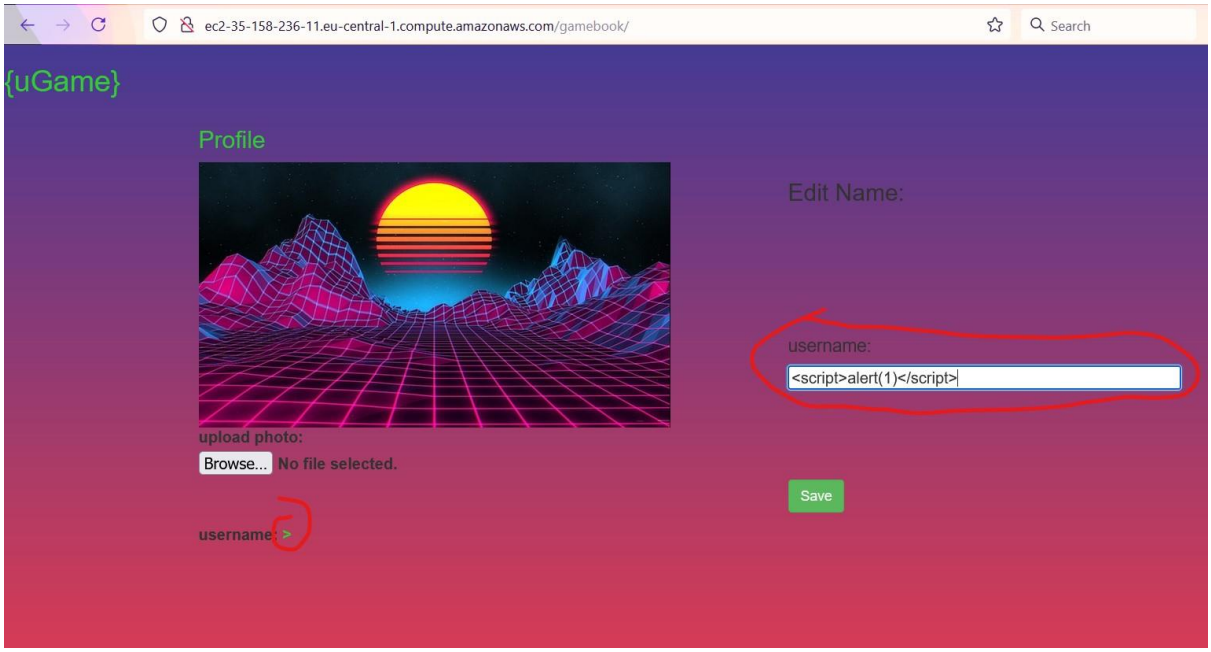
we are creating a new social media app for gaming , make sure its secure enough.

Link: http://ec2-35-158-236-11.eu-central-1.compute.amazonaws.com/gamebook/

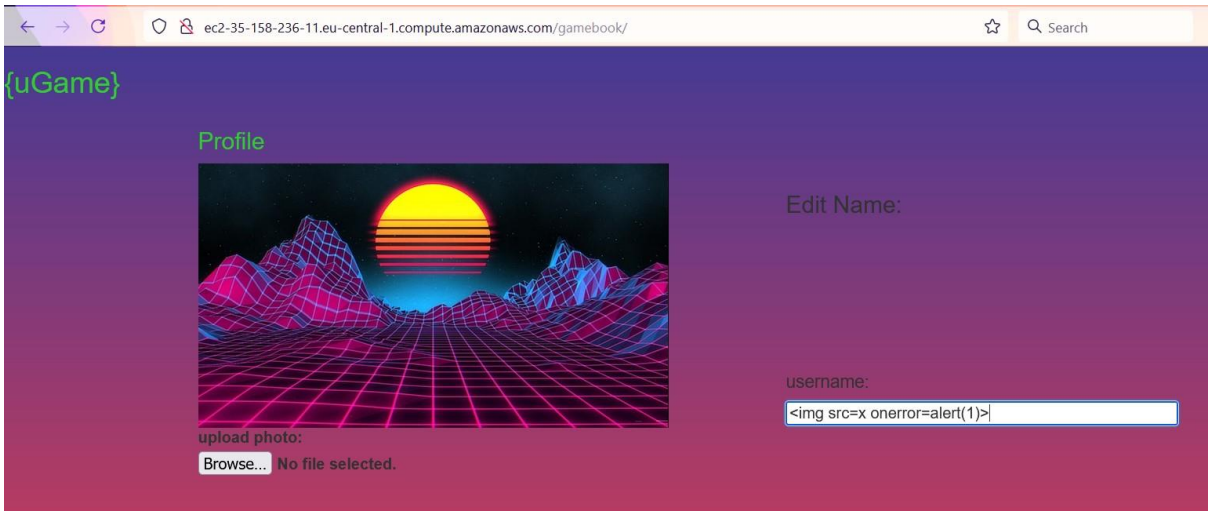
The next challenge was called Ugame which had a description of "**we are creating a new social media app for gaming, make sure its secure enough**" going to the link that was provided I found it was a website with only one entry point which is the username area. I typed in the test and the website reflected the name on the page as shown below.



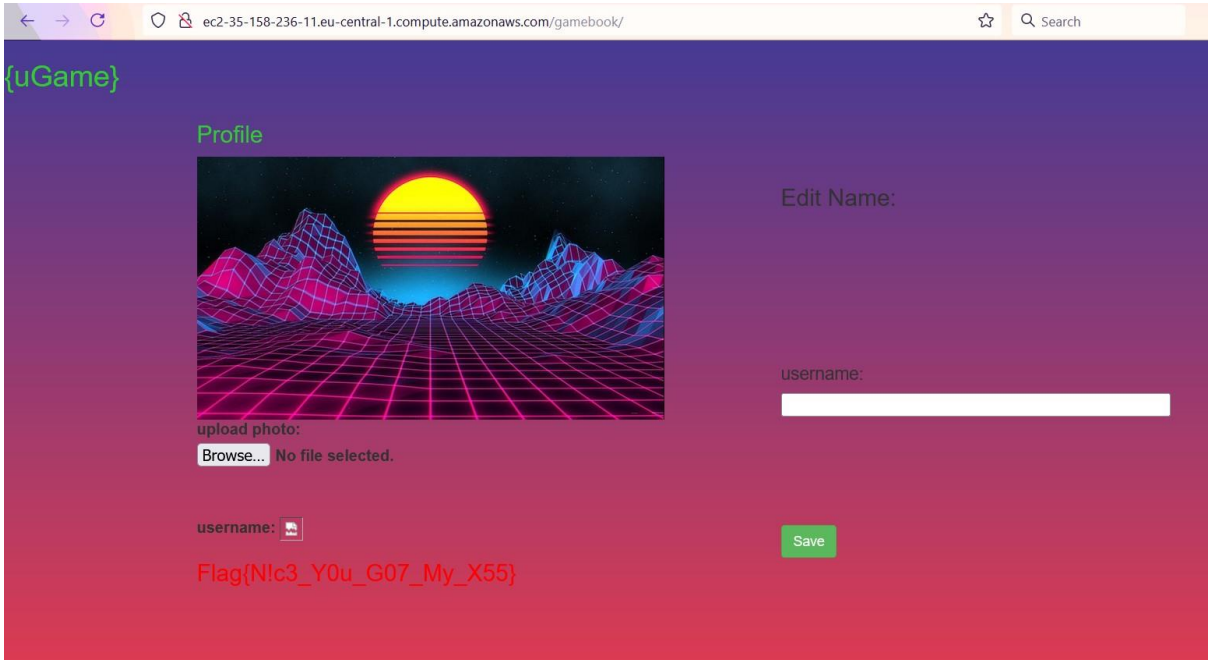
Next, I tried to perform XSS to see how the website would react, I inserted the payload and sent the request



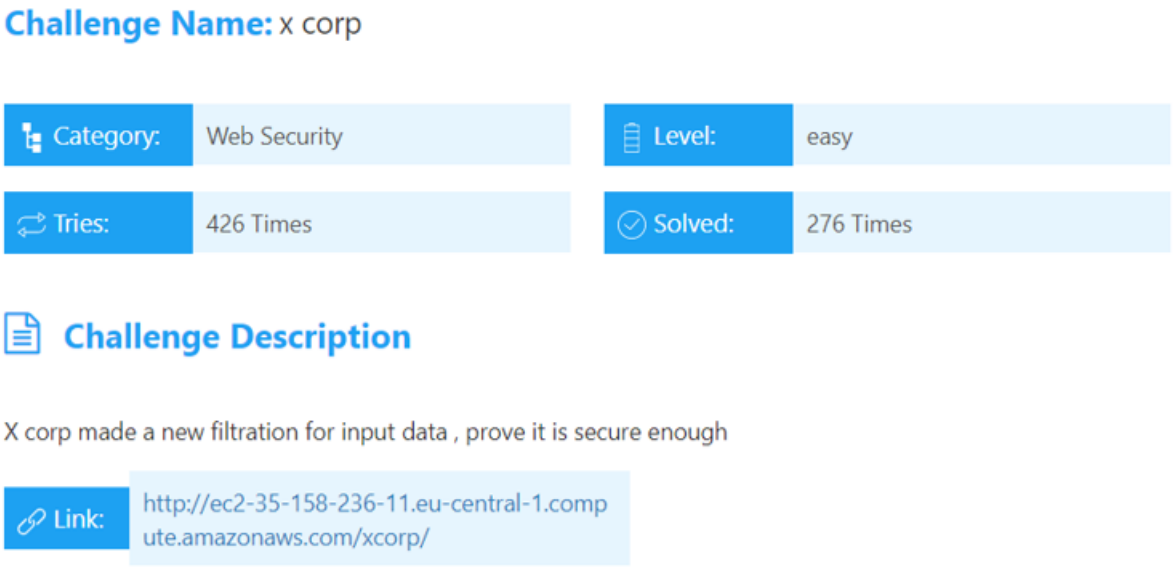
And from the result seems that there is a filter in the backend that's which filtered the word script, so we need to find a bypass of the filter which was the one as shown below.



Upon execution of the new XSS payload I managed to get the flag as shown below.



20. X corp

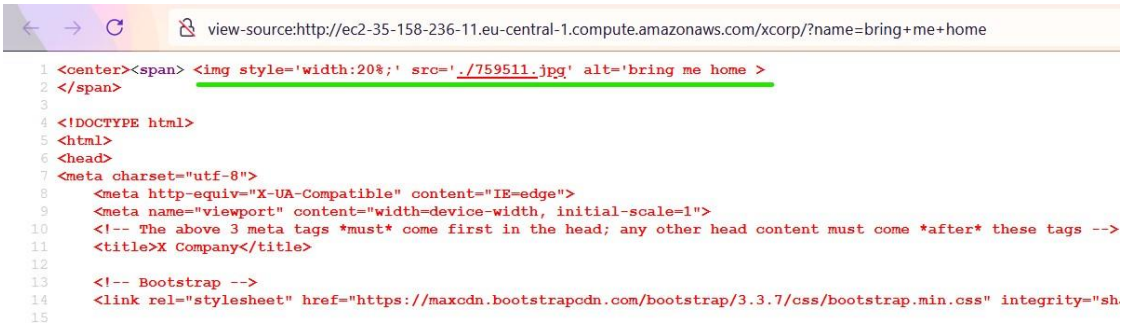


The next challenge was called x corp which had a description of "**X corp made a new filtration for input data, prove it is secure enough**" going to the link that was provided I found it was a website looking around the website there wasn't anything useful, so I decided to play with the input.

I typed in bring me home to see what will happen

Sending the request, it returns the below image as shown

Viewing the source code, we notice that our input is replaced into the alt attribute for the img tag, and we can draw a conclusion that this could potentially be a XSS vulnerability.



i decided to send the following payload **test'onload=alert(1)** The onload attribute is an event handler which will execute the alert(1) on the loading of the html page and after sending the payload we got the flag.