<u>Introduction to Offensive Security CTF tasks:</u>

In this assignment, we are asked to solve questions from the picoctf and hackucf challenges.

CTF stands for Capture The Flag. In the realm of cybersecurity, a CTF is a competition or challenge that involves various tasks centered around solving security-related problems. Participants, often cybersecurity enthusiasts, hackers, or professionals, engage in these challenges to demonstrate their skills in different areas of security.

Core Aspects of CTFs:

1. Problem-Solving: CTFs present a series of puzzles, challenges, or real-life scenarios related to cybersecurity. These challenges can cover a wide range of topics including cryptography, reverse engineering, web security, binary exploitation, forensics, and more.

2. Teamwork or Individual Effort: Participants can engage in CTFs either individually or as part of a team. Team-based competitions encourage collaboration and the pooling of diverse skill sets to solve challenges efficiently.

3. Diversity in Challenges: Challenges vary in difficulty levels, requiring participants to apply their knowledge in coding, networking, system administration, and other technical skills to find hidden vulnerabilities or solve puzzles.

4. Time Constraints: CTFs often have a time limit, intensifying the challenge and encouraging quick thinking and problem-solving skills under pressure.

5. Learning Opportunity: CTFs serve as an excellent learning platform. Participants can gain hands-on experience, learn new techniques, and improve their cybersecurity skills in a safe and controlled environment.

Types of CTF Challenges:

Forensics: Analyzing digital artifacts, file formats, logs, or network captures to extract hidden information.

Web Security: Identifying and exploiting vulnerabilities in web applications or servers.

Reverse Engineering: Disassembling and analyzing software or firmware to understand its functionality or find vulnerabilities.

Cryptography: Decrypting or breaking encoded messages, hashing algorithms, or encryption methods.

Binary Exploitation: Understanding and exploiting vulnerabilities in compiled binaries or executables.

CTFs are not only competitive events but also serve as a means for skill development, knowledge sharing, and networking within the cybersecurity community. They often attract participants from diverse backgrounds who are eager to learn, innovate, and excel in the field of cybersecurity.

I used kali linux for most of these tasks and since I have never used kali before, it was a little tricky for me to understand.

The table given below shows what challenges I did, which website I used and the points I earned in total:

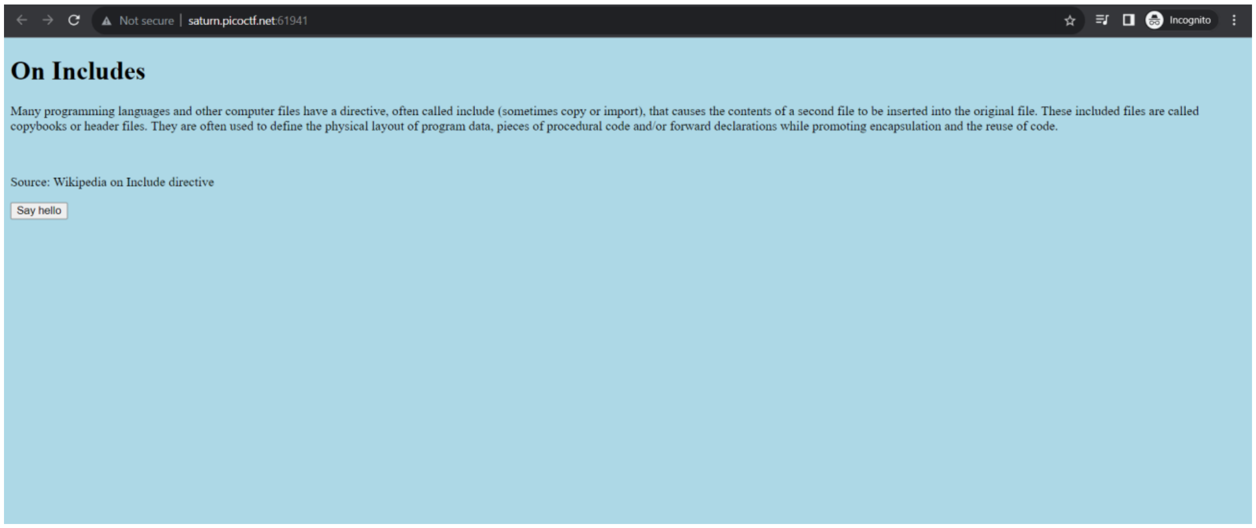
CTF Site Name	Name of Challenge	Category	Scoreboard Points	Assignment Points
picoCTF	***Includes***	*Web Exploitation*	100	10
	Inspect HTML	*Web Exploitation*	100	10
	CVE-XXXX-XXXX	*Binary Exploitation*	100	10
	buffer overflow 0	*Buffer Overflow*	100	10

	search source	*Web Exploitation*	100	10
	Forbidden Paths	*Web Exploitation*	200	20
	Power Cookie	*Web Exploitation*	200	20
	Roboto Sans	Web Exploitation	200	20
	Secrets	Web Exploitation	200	20
Total			1300	130

The table is not clear enough hence I am adding the excel sheet along with this in the submission. It also contains the comments associated with all the challenges.

Task 1: Includes by PICOCTF (100 points):

This challenge was based on web exploitation and it was a little tricky in the beginning to understand. The following screenshot highlights the challenge:

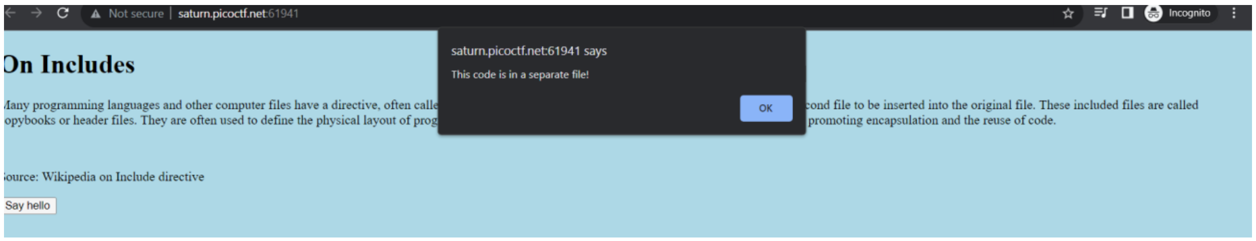


On Includes

Many programming languages and other computer files have a directive, often called include (sometimes copy or import), that causes the contents of a second file to be inserted into the original file. These included files are called copybooks or header files. They are often used to define the physical layout of program data, pieces of procedural code and/or forward declarations while promoting encapsulation and the reuse of code.

Source: Wikipedia on Include directive

When I clicked say hello, I got the following result:



Not secure | saturn.picoctf.net:61941

On Includes

lany programming languages and other computer files have a directive, often calle
opybooks or header files. They are often used to define the physical layout of prog

saturn.picoctf.net:61941 says

This code is in a separate file!

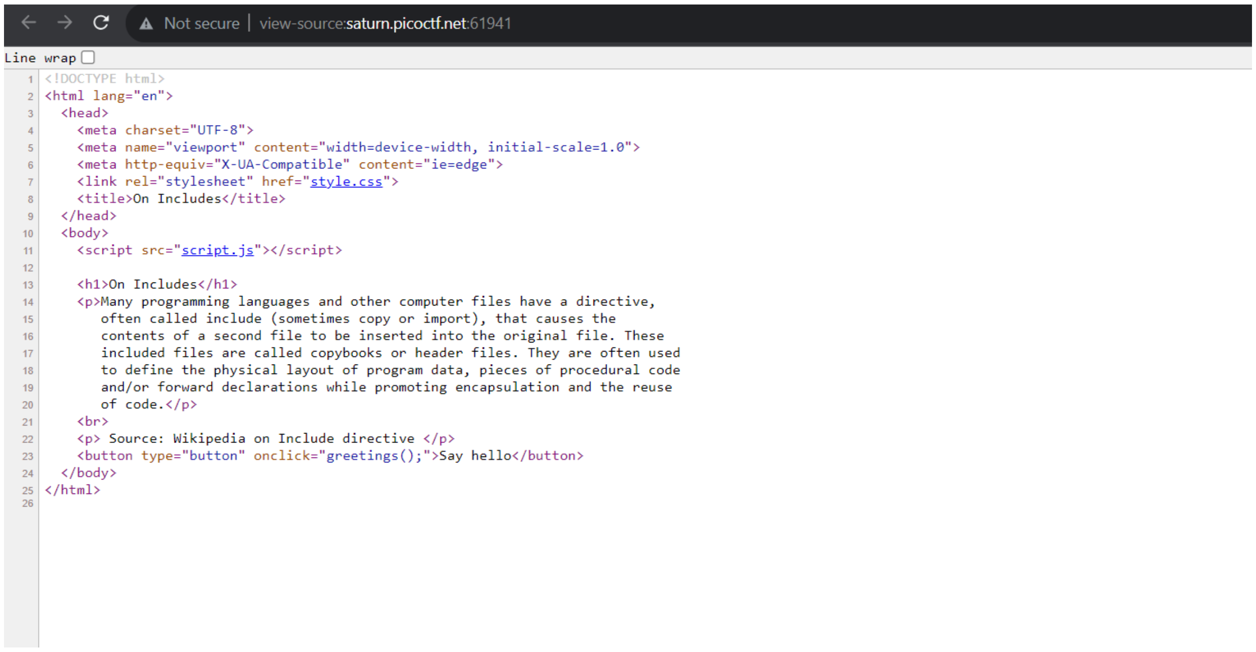
OK

cond file to be inserted into the original file. These included files are called
promoting encapsulation and the reuse of code.

ource: Wikipedia on Include directive

Say hello

So, the first thing we do when we are attempting a web exploitation challenge is view the page resource:

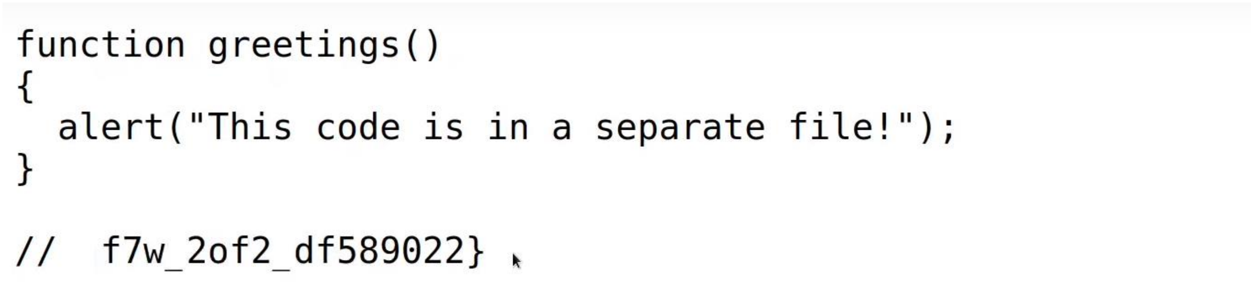


Not secure | view-source:saturn.picoctf.net:61941

Line wrap

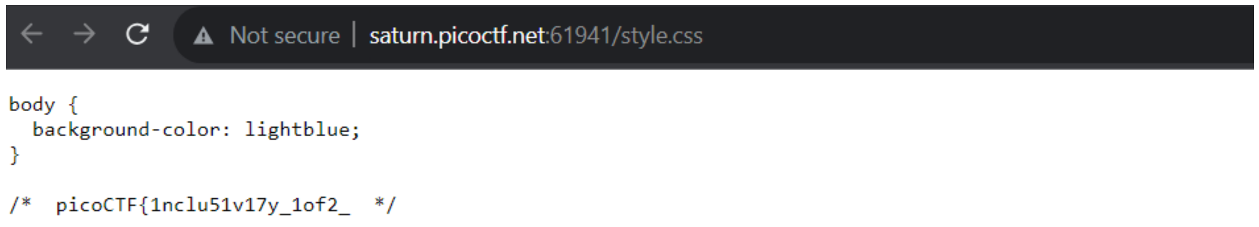
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style.css">
    <title>On Includes</title>
  </head>
  <body>
    <script src="script.js"></script>

    <h1>On Includes</h1>
    <p>Many programming languages and other computer files have a directive,
       often called include (sometimes copy or import), that causes the
       contents of a second file to be inserted into the original file. These
       included files are called copybooks or header files. They are often used
       to define the physical layout of program data, pieces of procedural code
       and/or forward declarations while promoting encapsulation and the reuse
       of code.</p>
    <br>
    <p> Source: Wikipedia on Include directive </p>
    <button type="button" onclick="greetings();">Say hello</button>
  </body>
</html>
```

When we analyze the above code, we can see that on pressing the say hello button, it goes to a function greetings() which is included in script.js. The paragraph was telling us the exact same thing about directive. Now when I looked at the script.js, I found out the following:



```javascript
function greetings()
{
  alert("This code is in a separate file!");
}

//  f7w_2of2_df589022}
```

This is what was stored in the script.js and as we can see, the comment part contains something of use to us. As we can see, this is actually part of the flag so I saved this part. Then I noticed that this HTML code also included a file, style.css when I opened this file, I got the following:



```
body {
  background-color: lightblue;
}

/*  picoCTF{1nclu51v17y_1of2_  */
```

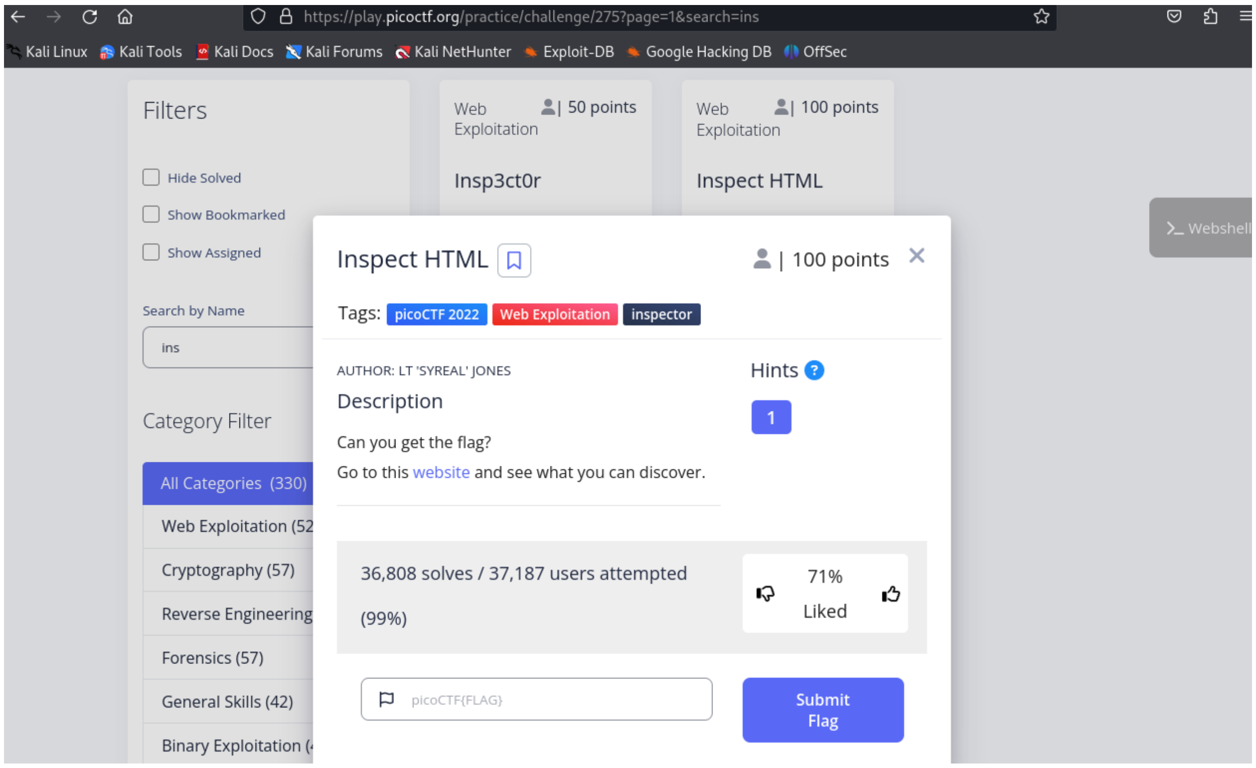
So, this is actually the first part of the flag and this is also included in a comment section, merging both of these I got the flag:

picoCTF{1nclu51v17y_1of2_f7w_2of2_6edef411}

By submitting this flag, I earned 100 points.

Task 2: Inspect HTML by PICOCTF(100 points):

This task is also based on web exploitation and is part of the picoctf 2022 challenge. The screenshot below shows the task in hand:


https://play.picoctf.org/practice/challenge/275?page=1&search=ins

Inspect HTML | 100 points

Tags: picoCTF 2022 Web Exploitation inspector

AUTHOR: LT 'SYREAL' JONES

Description

Can you get the flag?

Go to this website and see what you can discover.

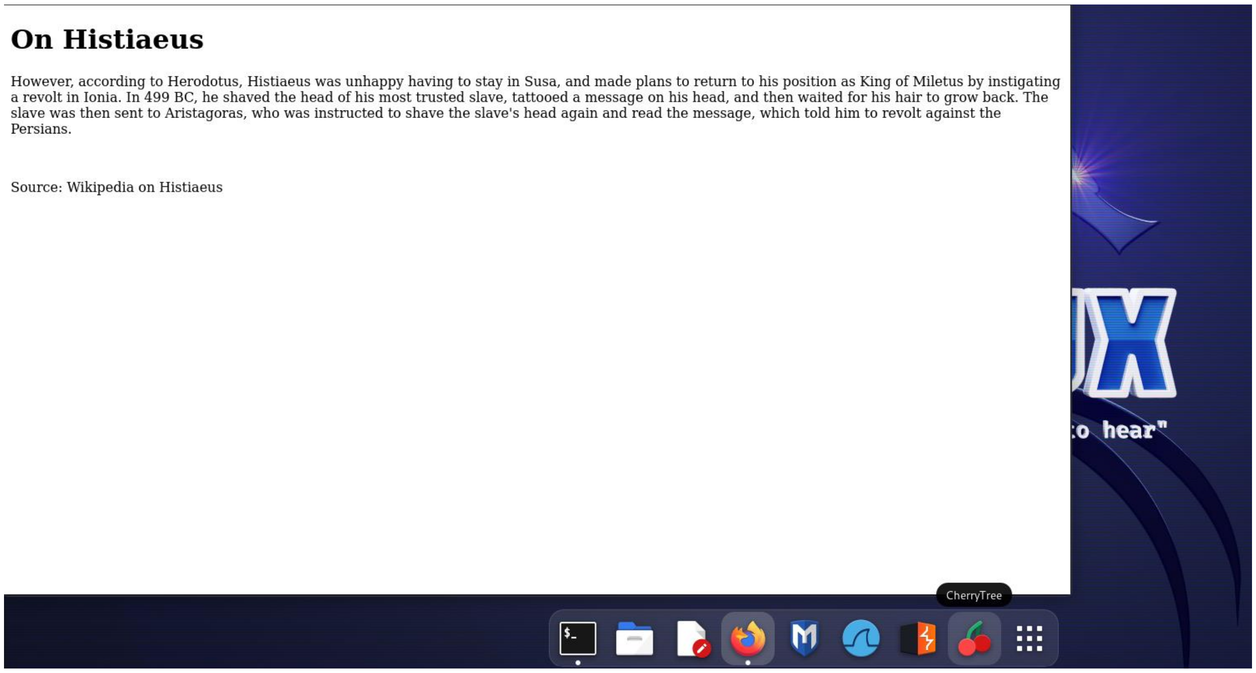
Hints 1

36,808 solves / 37,187 users attempted (99%)

71% Liked

picoCTF{FLAG}

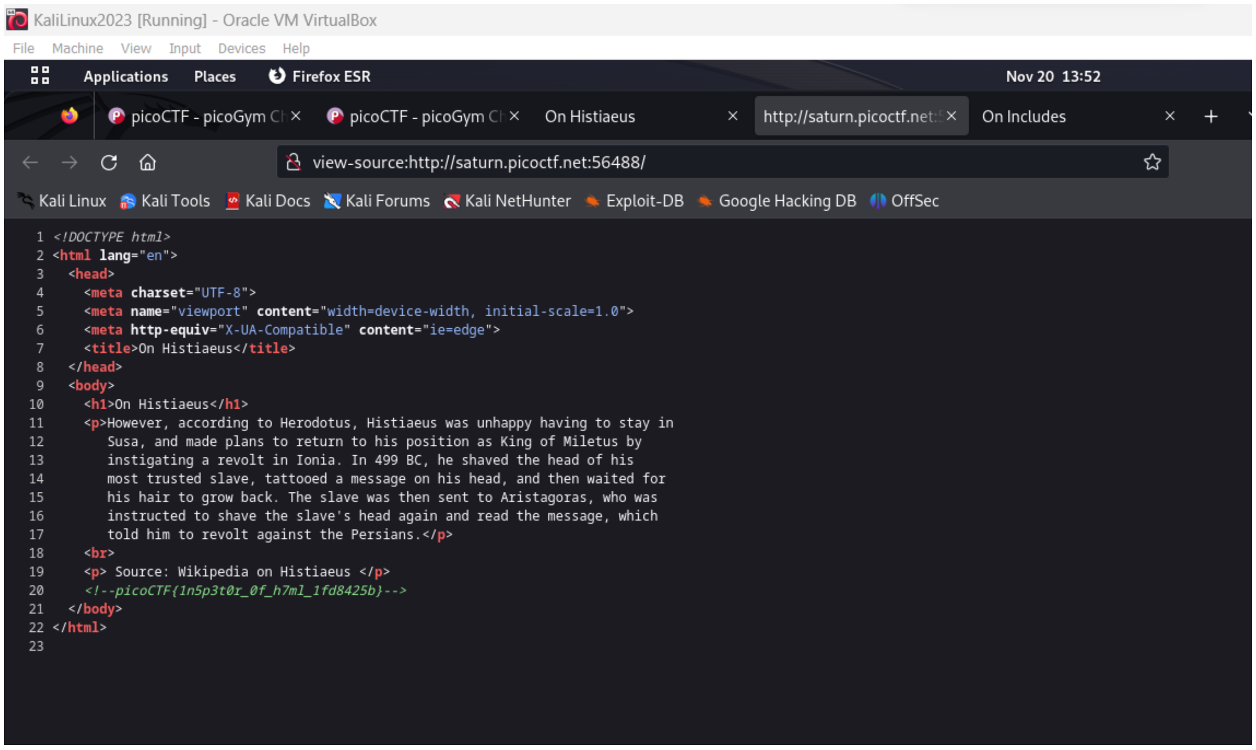
Submit Flag


On Histiaeus

However, according to Herodotus, Histiaeus was unhappy having to stay in Susa, and made plans to return to his position as King of Miletus by instigating a revolt in Ionia. In 499 BC, he shaved the head of his most trusted slave, tattooed a message on his head, and then waited for his hair to grow back. The slave was then sent to Aristagoras, who was instructed to shave the slave's head again and read the message, which told him to revolt against the Persians.

Source: Wikipedia on Histiaeus

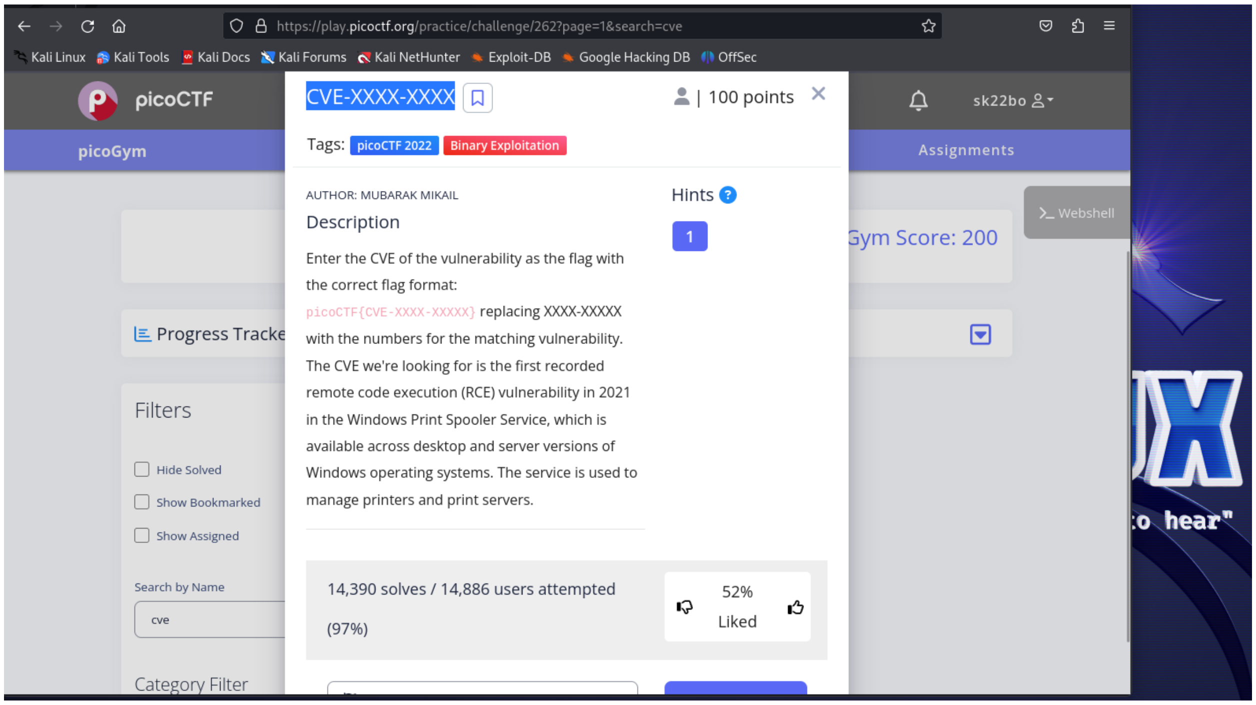
Next, as required I saw the source code of the webpage and found the following result:


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>On Histiaeus</title>
  </head>
  <body>
    <h1>On Histiaeus</h1>
    <p>However, according to Herodotus, Histiaeus was unhappy having to stay in
       Susa, and made plans to return to his position as King of Miletus by
       instigating a revolt in Ionia. In 499 BC, he shaved the head of his
       most trusted slave, tattooed a message on his head, and then waited for
       his hair to grow back. The slave was then sent to Aristagoras, who was
       instructed to shave the slave's head again and read the message, which
       told him to revolt against the Persians.</p>
    <br>
    <p> Source: Wikipedia on Histiaeus </p>
    <!--picoCTF{1n5p3t0r_0f_h7ml_1fd8425b}-->
  </body>
</html>
```

As we can see in the above source code, the flag is included in the comment under the <p> Source tab. This was the easiest challenge of all because we didn't really do anything apart from viewing the webpage in source code format and we successfully found the flag.

Task 3: CVE-XXXX-XXXX by PICOCTF(100 points):

This task is also part of the PICOCTF challenge and it is a binary exploitation challenge. The screenshot below gives us an idea on what the challenge is:


https://play.picoctf.org/practice/challenge/262?page=1&search=cve

CVE-XXXX-XXXX | 100 points

Tags: picoCTF 2022 Binary Exploitation

AUTHOR: MUBARAK MIKAIL

Description

Enter the CVE of the vulnerability as the flag with the correct flag format:

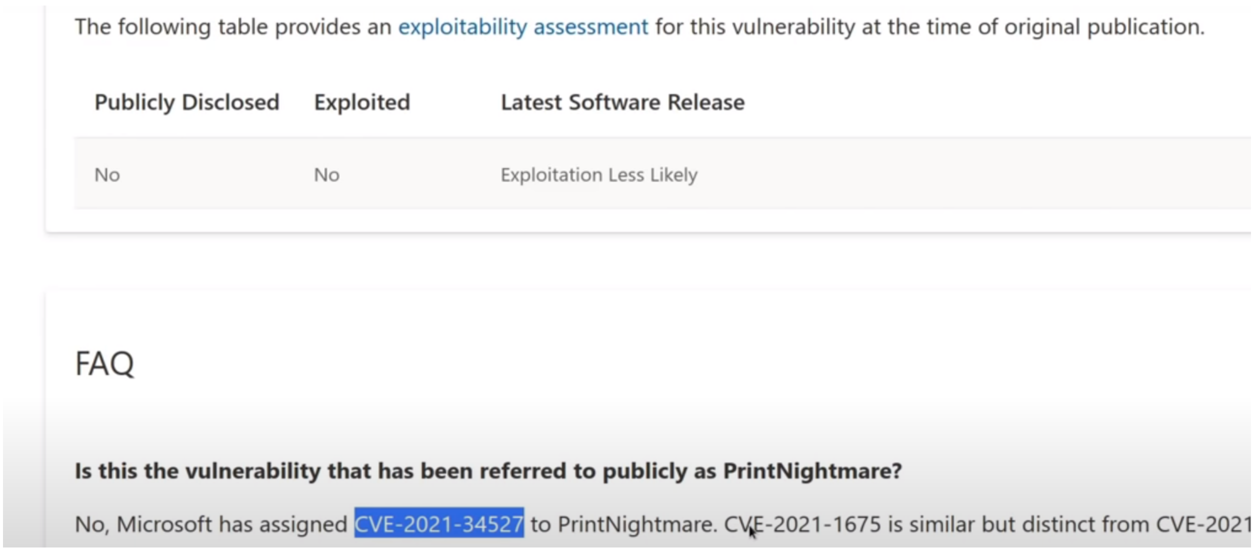
`picoCTF{CVE-XXXX-XXXXX}` replacing XXXX-XXXXX with the numbers for the matching vulnerability.

The CVE we're looking for is the first recorded remote code execution (RCE) vulnerability in 2021 in the Windows Print Spooler Service, which is available across desktop and server versions of Windows operating systems. The service is used to manage printers and print servers.

14,390 solves / 14,886 users attempted (97%)

52% Liked

I noticed that this is one of the Microsoft vulnerabilities in 2021. It was most popularly known as 2021 windows printer exploit. I went through the Microsoft website and found the flag in the form of CVE. The screenshot below is taken from the Microsoft website:


The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release
No	No	Exploitation Less Likely

FAQ

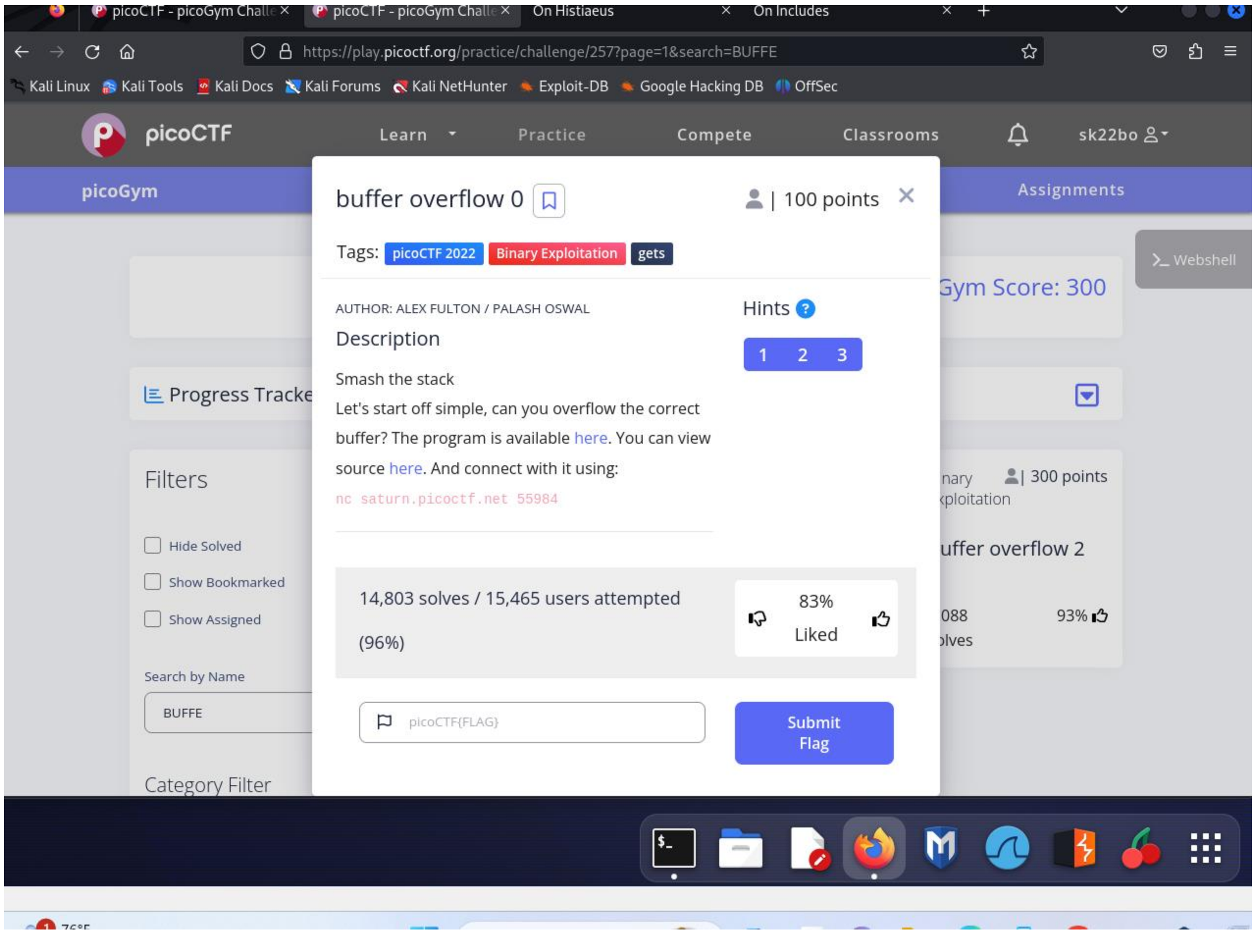
Is this the vulnerability that has been referred to publicly as PrintNightmare?

No, Microsoft has assigned CVE-2021-34527 to PrintNightmare. CVE-2021-1675 is similar but distinct from CVE-2021

Hence, I just had to write this number in the picoCTF format. This was the flag for this challenge. And it was successfully captured.

Task 4: buffer overflow 0 by PICOCTF(100 points):

This challenge is associated with buffer overflow attack. The screenshot below gives a description of the challenge:



buffer overflow 0

100 points

Tags: picoCTF 2022 Binary Exploitation gets

AUTHOR: ALEX FULTON / PALASH OSWAL

Description

Smash the stack

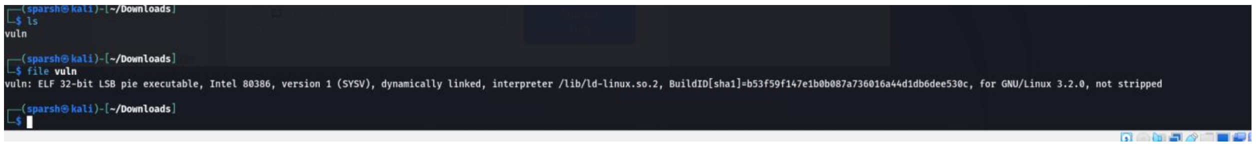
Let's start off simple, can you overflow the correct buffer? The program is available here. You can view source here. And connect with it using:

```
nc saturn.picoctf.net 55984
```

14,803 solves / 15,465 users attempted (96%)

83% Liked

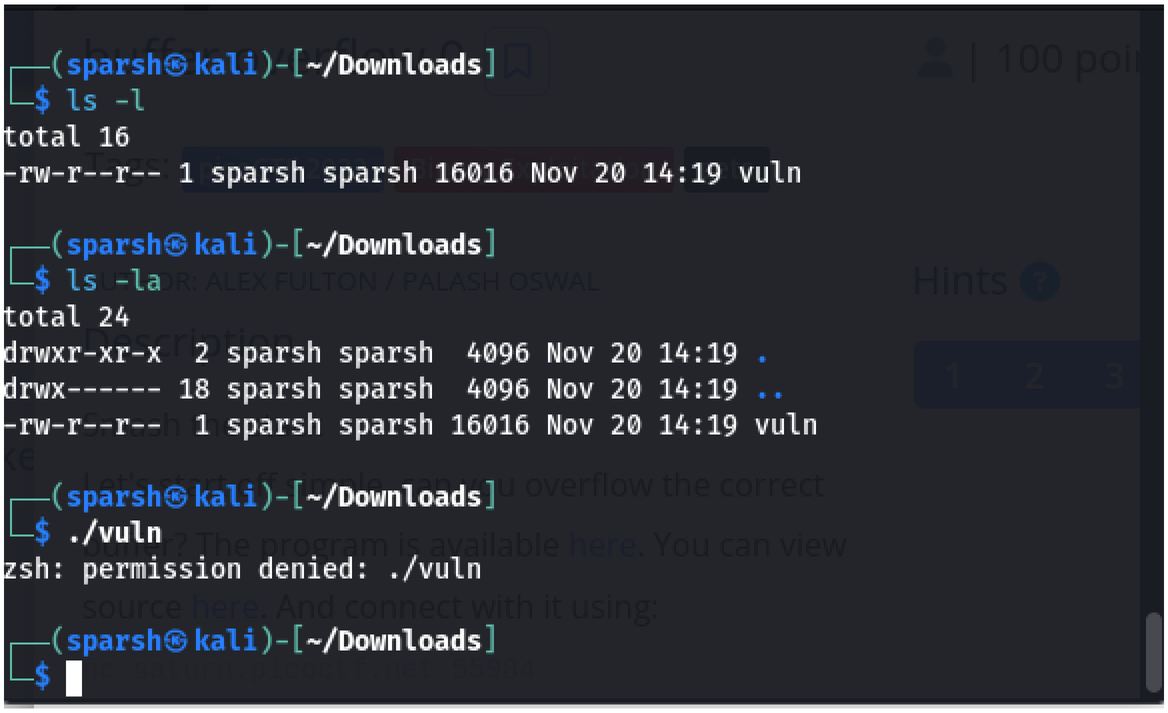
After this I used kali to access the vuln file that was downloaded using the links in this challenge:



```
┌──(sparsh㉿kali)-[~/Downloads]
└─$ ls
vuln

┌──(sparsh㉿kali)-[~/Downloads]
└─$ file vuln
vuln: ELF 32-bit LSB pie executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1]=b53f59f147e1b0b087a736016a44d1db6dee530c, for GNU/Linux 3.2.0, not stripped

┌──(sparsh㉿kali)-[~/Downloads]
└─$ 
```

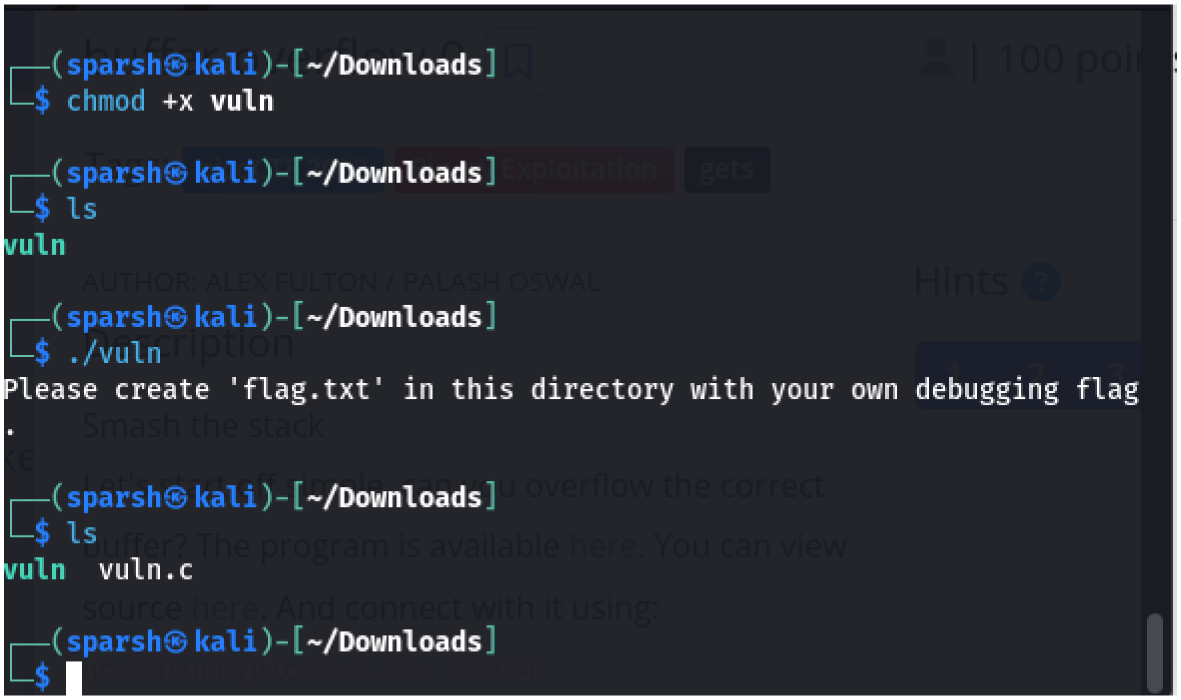
The screenshot above shows the successful access of vuln file to see its properties using kali linux file function.

And when I tried running this program, it was non executable:



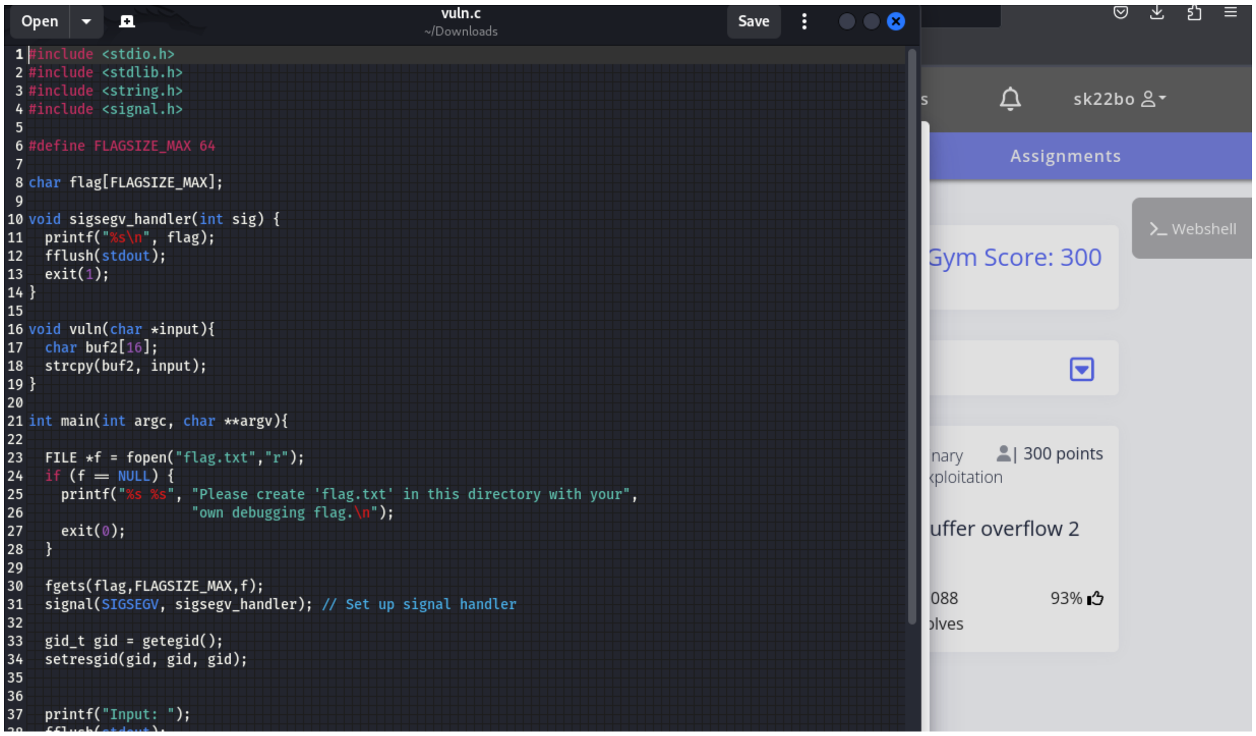
```
┌──(sparsh㉿kali)-[~/Downloads]
└─$ ls -l
total 16
-rw-r--r-- 1 sparsh sparsh 16016 Nov 20 14:19 vuln

┌──(sparsh㉿kali)-[~/Downloads]
└─$ ls -la
total 24
drwxr-xr-x  2 sparsh sparsh  4096 Nov 20 14:19 .
drwx------ 18 sparsh sparsh  4096 Nov 20 14:19 ..
-rw-r--r--  1 sparsh sparsh 16016 Nov 20 14:19 vuln

┌──(sparsh㉿kali)-[~/Downloads]
└─$ ./vuln
zsh: permission denied: ./vuln

┌──(sparsh㉿kali)-[~/Downloads]
└─$ 
```

After this I used the chmod +x function with vuln and I also downloaded the vuln.c file, the screenshot below shows my execution:

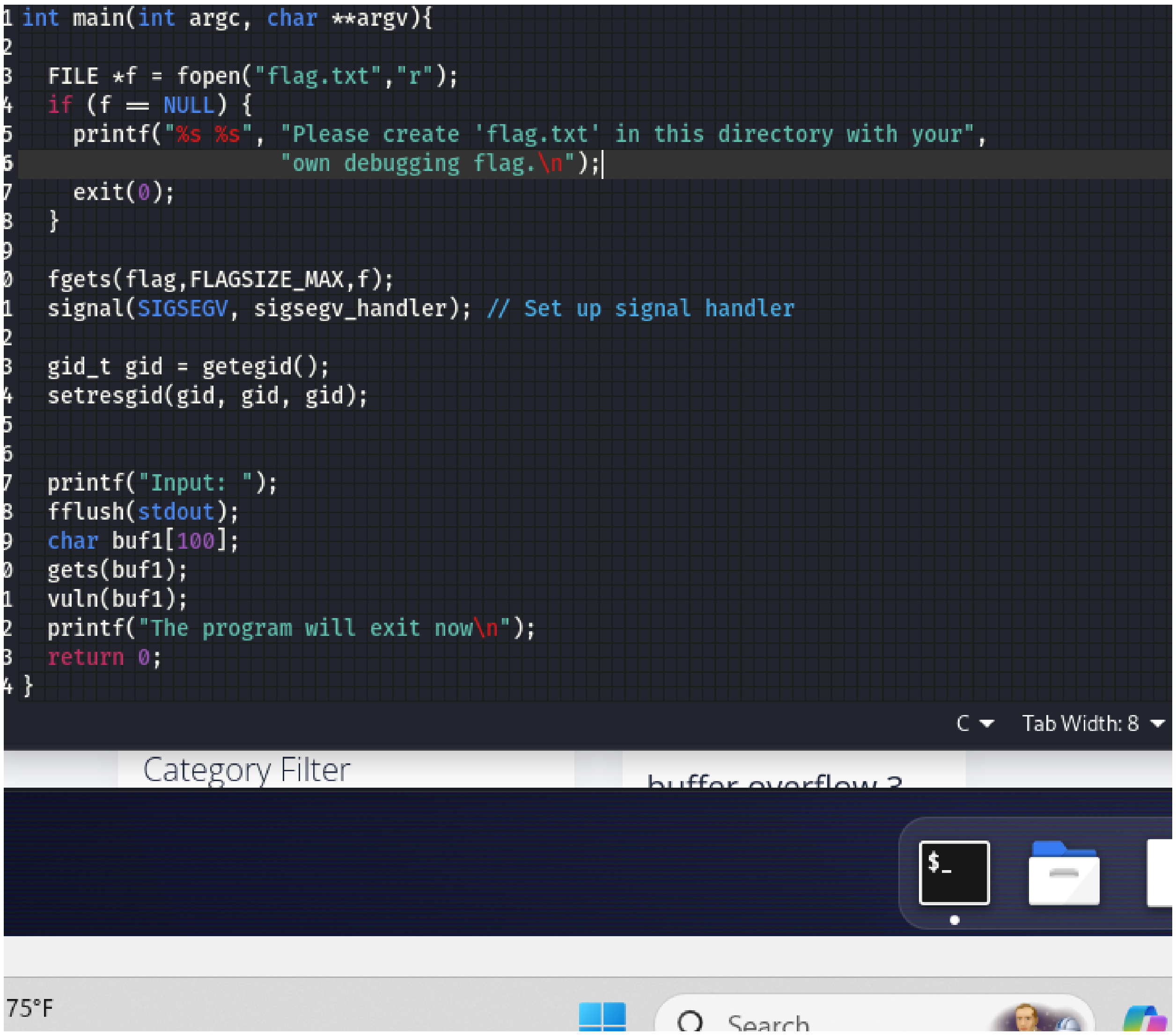


```
┌──(sparsh㉿kali)-[~/Downloads]
└─$ chmod +x vuln

┌──(sparsh㉿kali)-[~/Downloads]
└─$ ls
vuln

┌──(sparsh㉿kali)-[~/Downloads]
└─$ ./vuln
Please create 'flag.txt' in this directory with your own debugging flag
.

┌──(sparsh㉿kali)-[~/Downloads]
└─$ ls
vuln  vuln.c

┌──(sparsh㉿kali)-[~/Downloads]
└─$ 
```

The vuln.c file contained the following code:

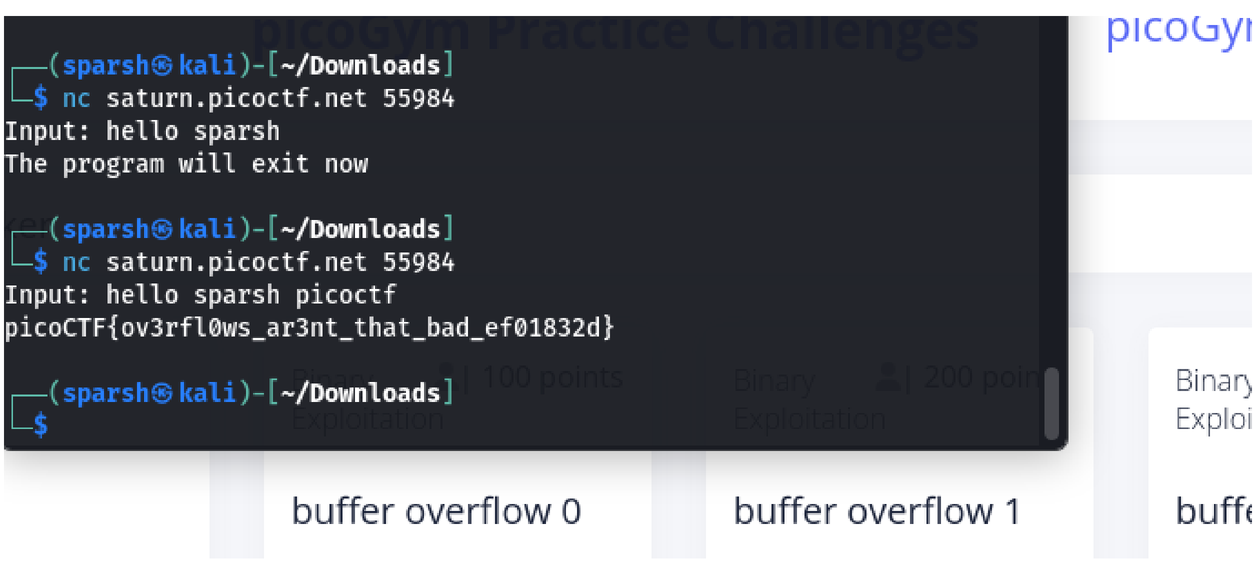


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>

#define FLAGSIZE_MAX 64

char flag[FLAGSIZE_MAX];

void sigsegv_handler(int sig) {
  printf("%s\n", flag);
  fflush(stdout);
  exit(1);
}

void vuln(char *input){
  char buf2[16];
  strcpy(buf2, input);
}

int main(int argc, char **argv){

  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("%s %s", "Please create 'flag.txt' in this directory with your",
                    "own debugging flag.\n");
    exit(0);
  }

  fgets(flag,FLAGSIZE_MAX,f);
  signal(SIGSEGV, sigsegv_handler); // Set up signal handler

  gid_t gid = getegid();
  setresgid(gid, gid, gid);


  printf("Input: ");
```


```c
int main(int argc, char **argv){

  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("%s %s", "Please create 'flag.txt' in this directory with your",
                    "own debugging flag.\n");
    exit(0);
  }

  fgets(flag,FLAGSIZE_MAX,f);
  signal(SIGSEGV, sigsegv_handler); // Set up signal handler

  gid_t gid = getegid();
  setresgid(gid, gid, gid);


  printf("Input: ");
  fflush(stdout);
  char buf1[100];
  gets(buf1);
  vuln(buf1);
  printf("The program will exit now\n");
  return 0;
}
```

After this, I started analyzing the code and I saw that this code uses a gets function. gets() is a function that reads a line from stdin into the buffer points to by s until either a terminating newline or E0F, which it replaces with a null byte. No check for buffer overrun is performed. Even when I tried running the gcc command, it got the warning that the gets function is dangerous and should not be used.

Now, I grabbed the remote connection string as given on the picoCTF website and implemented it on kali linux, first I inserted a message hello sparsh but since it is less than 16 characters, the program exited but as soon as I surpassed the buffer limit, I got the following result:



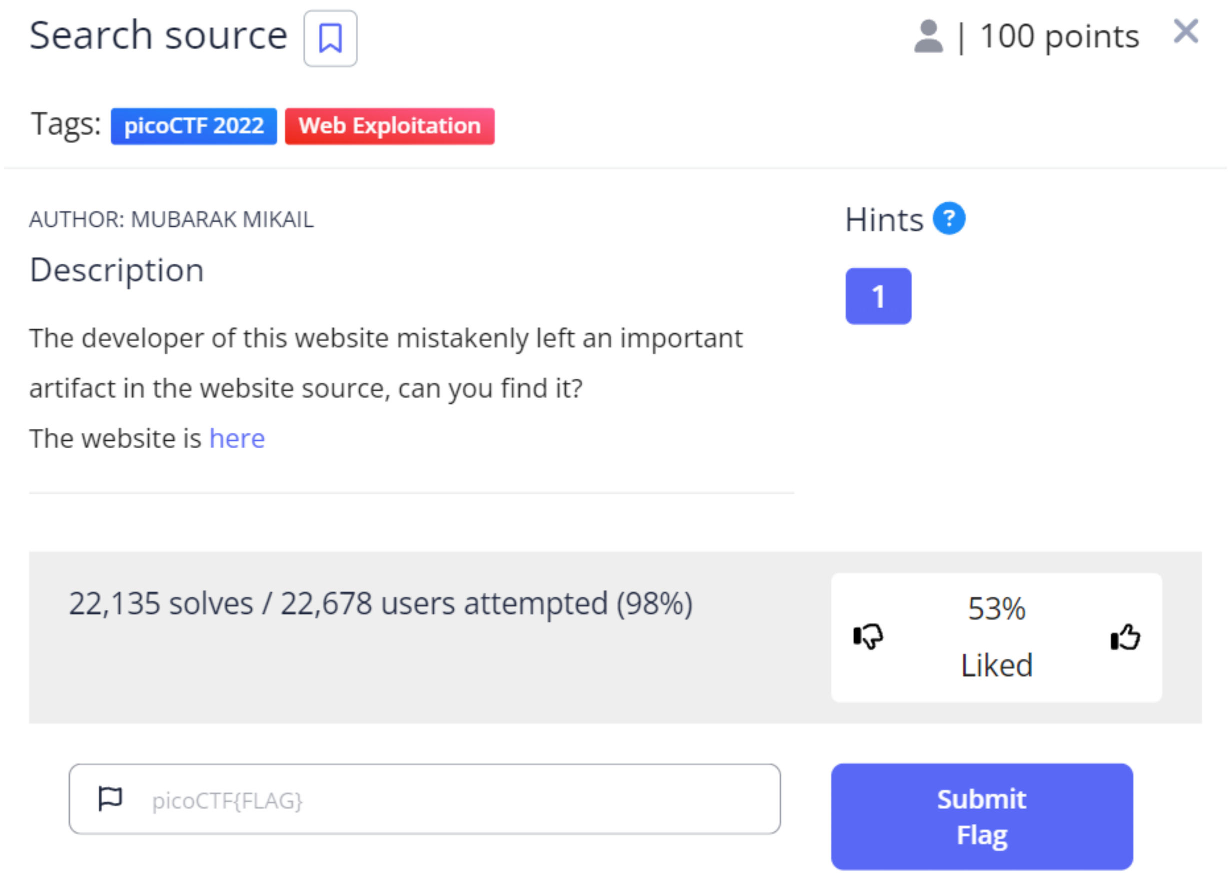
```
┌──(sparsh㉿kali)-[~/Downloads]
└─$ nc saturn.picoctf.net 55984
Input: hello sparsh
The program will exit now

┌──(sparsh㉿kali)-[~/Downloads]
└─$ nc saturn.picoctf.net 55984
Input: hello sparsh picoctf
picoCTF{ov3rfl0ws_ar3nt_that_bad_ef01832d}

┌──(sparsh㉿kali)-[~/Downloads]
└─$
```

As we can see, we acquired the flag as soon as we overflowed the buffer. This is a classic buffer overflow exploitation example.

Hence, we were able to acquire the flag successfully by analyzing the code and overflowing the buffer.

Task 5: search source by picoCTF(100 points):

This task was also part of the picoCTF challenge and is based on Web Exploitation as well. The screenshot below gives an overview of the task:



Search source

100 points

Tags: picoCTF 2022 Web Exploitation

AUTHOR: MUBARAK MIKAIL

Description

The developer of this website mistakenly left an important artifact in the website source, can you find it?

The website is here

Hints 1

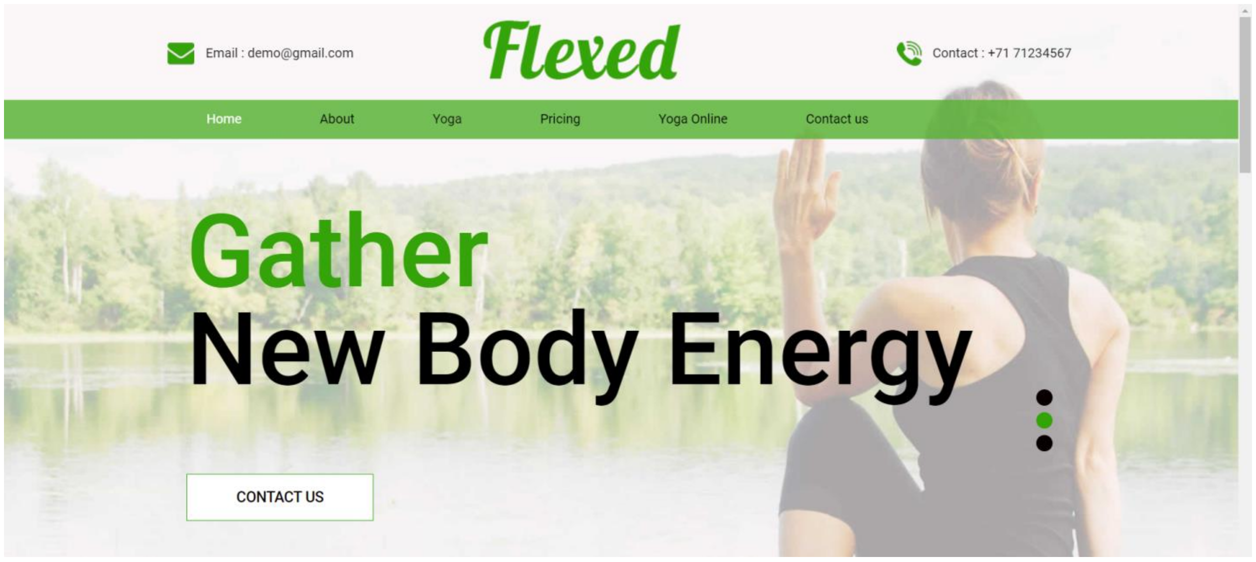
22,135 solves / 22,678 users attempted (98%)

53% Liked

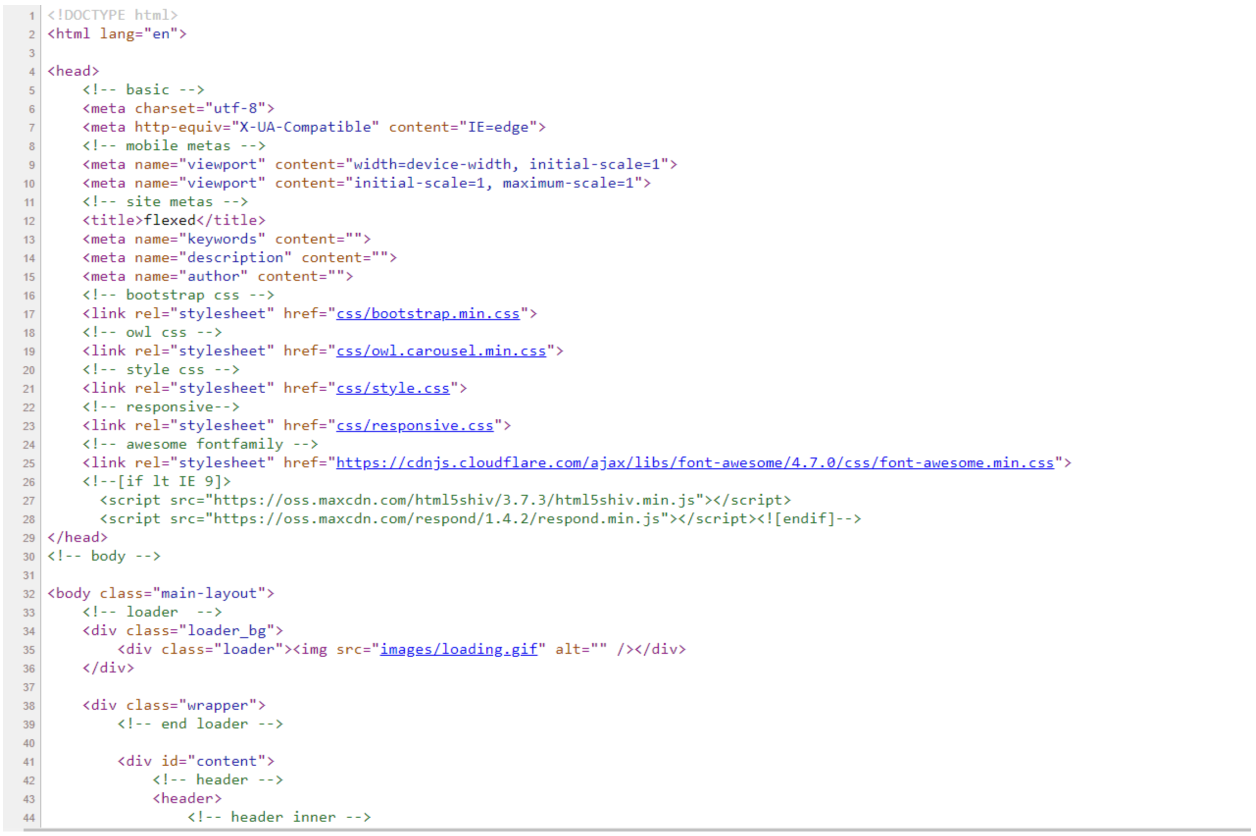
picoCTF{FLAG}

Submit Flag

And this is the website that was associated with this task:

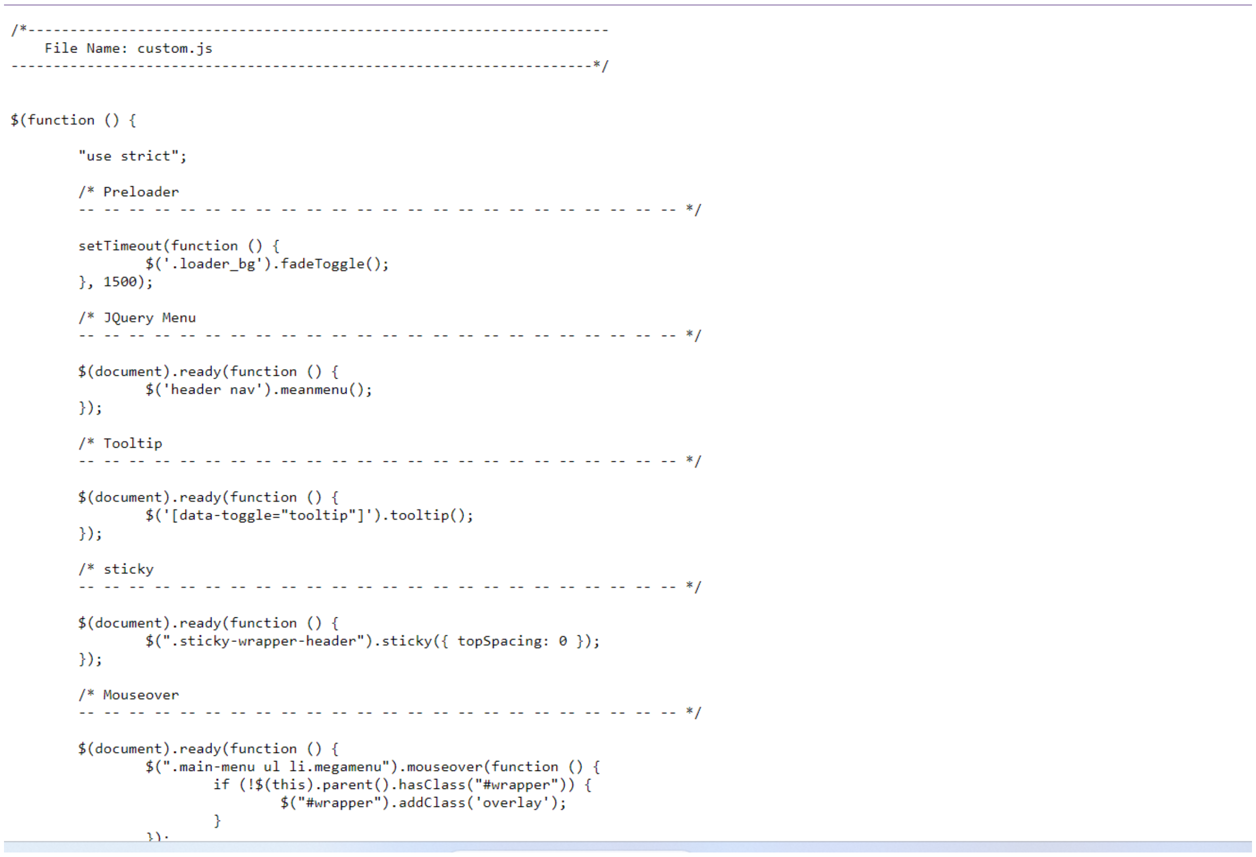


When I clicked the available options on the navigation bar, it just drops down to the webpage where their section starts. Next step as always is to view the source code:



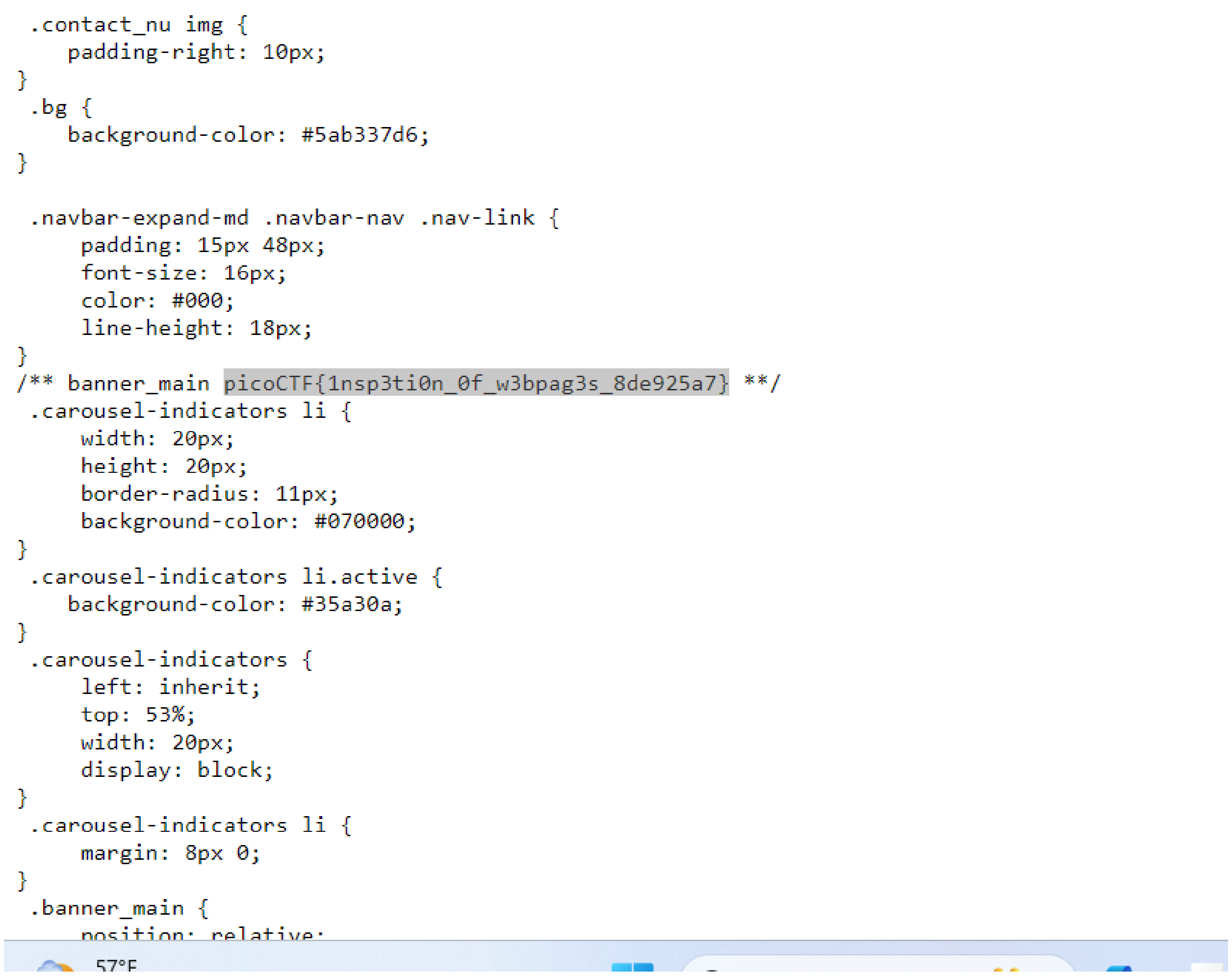
```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- basic -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- mobile metas -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <!-- site metas -->
    <title>flexed</title>
    <meta name="keywords" content="">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- bootstrap css -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <!-- owl css -->
    <link rel="stylesheet" href="css/owl.carousel.min.css">
    <!-- style css -->
    <link rel="stylesheet" href="css/style.css">
    <!-- responsive-->
    <link rel="stylesheet" href="css/responsive.css">
    <!-- awesome fontfamily -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <!--[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]-->
</head>
<!-- body -->

<body class="main-layout">
    <!-- loader  -->
    <div class="loader_bg">
        <div class="loader"><img src="images/loading.gif" alt="" /></div>
    </div>

    <div class="wrapper">
        <!-- end loader -->

        <div id="content">
            <!-- header -->
            <header>
                <!-- header inner -->
```

After this I started analyzing the javascript code and reached at customs.js which looked a little different:



```
/*--------------------------------------------------------------------
    File Name: custom.js
--------------------------------------------------------------------*/


$(function () {

        "use strict";

        /* Preloader
        -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- */

        setTimeout(function () {
                $('.loader_bg').fadeToggle();
        }, 1500);

        /* JQuery Menu
        -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- */

        $(document).ready(function () {
                $('header nav').meanmenu();
        });

        /* Tooltip
        -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- */

        $(document).ready(function () {
                $('[data-toggle="tooltip"]').tooltip();
        });

        /* sticky
        -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- */

        $(document).ready(function () {
                $(".sticky-wrapper-header").sticky({ topSpacing: 0 });
        });

        /* Mouseover
        -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- */

        $(document).ready(function () {
                $(".main-menu ul li.megamenu").mouseover(function () {
                        if (!$(this).parent().hasClass("#wrapper")) {
                                $("#wrapper").addClass('overlay');
                        }
```

But after analyzing this javascript I couldn’t find the flag. Then I started analyzing the css sheets and soon I found the flag in css/style.css:

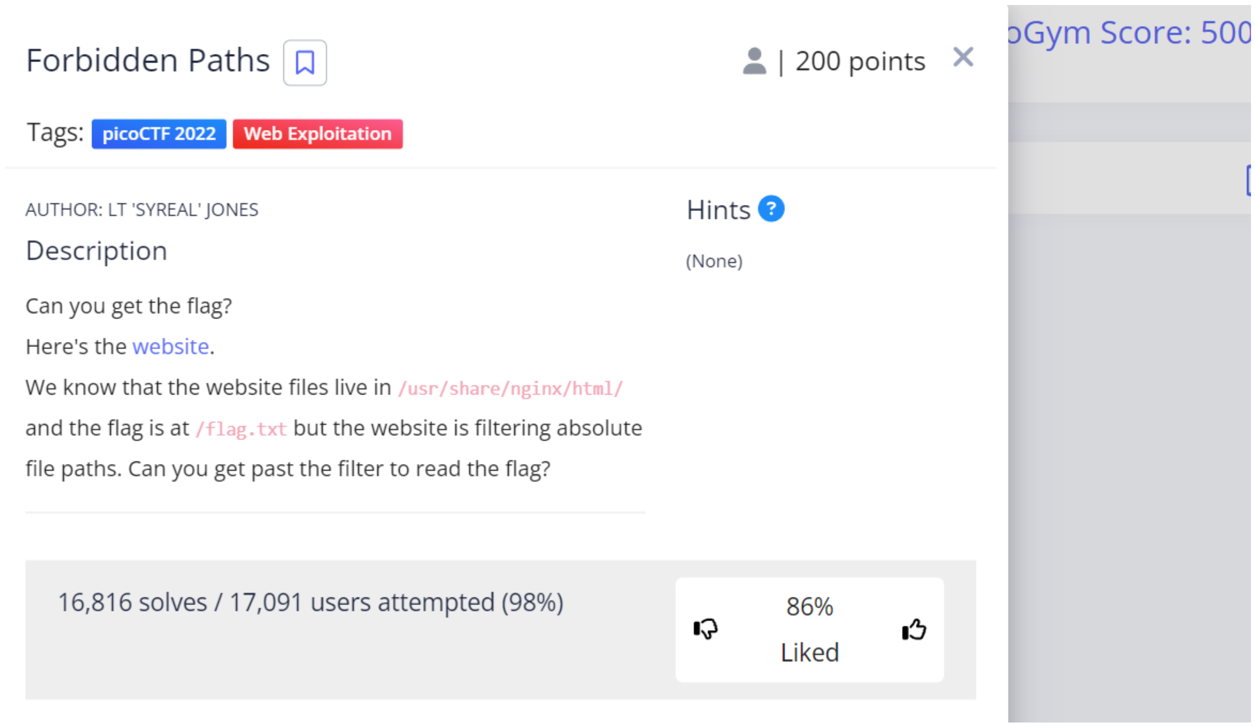


```
 .contact_nu img {
    padding-right: 10px;
}
 .bg {
    background-color: #5ab337d6;
}

 .navbar-expand-md .navbar-nav .nav-link {
     padding: 15px 48px;
     font-size: 16px;
     color: #000;
     line-height: 18px;
}
/** banner_main picoCTF{1nsp3ti0n_0f_w3bpag3s_8de925a7} **/
 .carousel-indicators li {
     width: 20px;
     height: 20px;
     border-radius: 11px;
     background-color: #070000;
}
 .carousel-indicators li.active {
    background-color: #35a30a;
}
 .carousel-indicators {
     left: inherit;
     top: 53%;
     width: 20px;
     display: block;
}
 .carousel-indicators li {
     margin: 8px 0;
}
 .banner_main {
     position: relative;
```

Hence, we successfully found the flag and this task was a success.

Task 6: Forbidden Paths picoCTF(200 points):

This task is also based on web exploitation. The screenshot below shows the task in hand:


Forbidden Paths

200 points

Tags: picoCTF 2022 Web Exploitation

AUTHOR: LT 'SYREAL' JONES

Description

Can you get the flag?
Here's the website.
We know that the website files live in `/usr/share/nginx/html/` and the flag is at `/flag.txt` but the website is filtering absolute file paths. Can you get past the filter to read the flag?

Hints

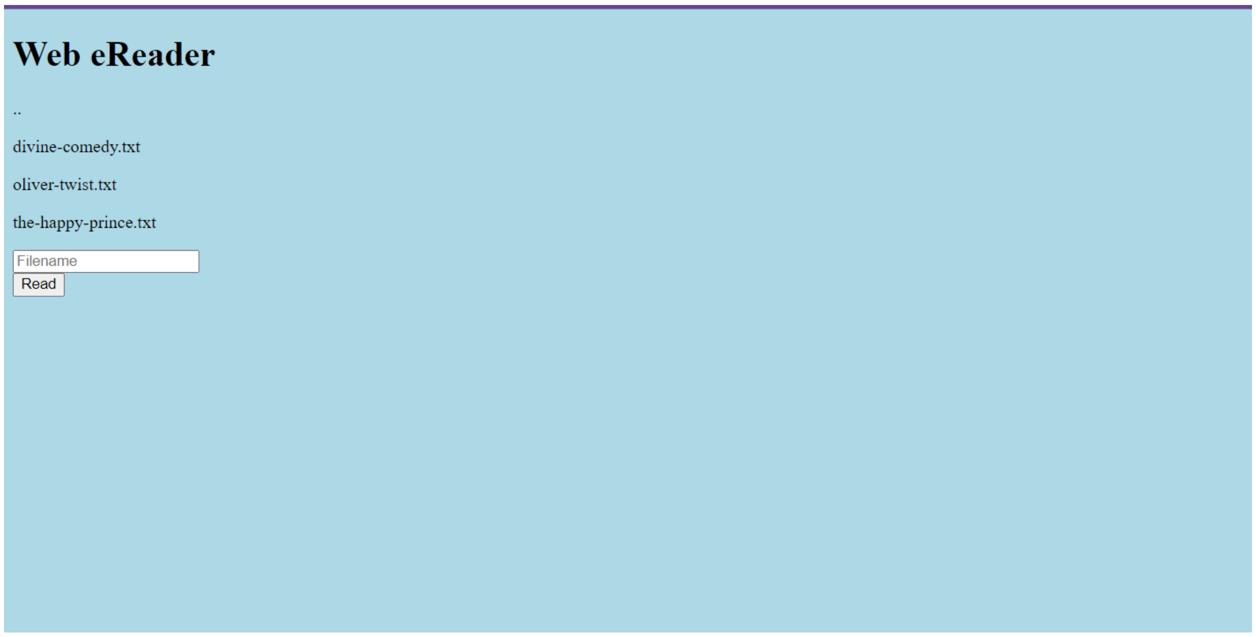
(None)

16,816 solves / 17,091 users attempted (98%)

86% Liked

oGym Score: 500

When I opened the website associated with this task, I got the following result:


Web eReader

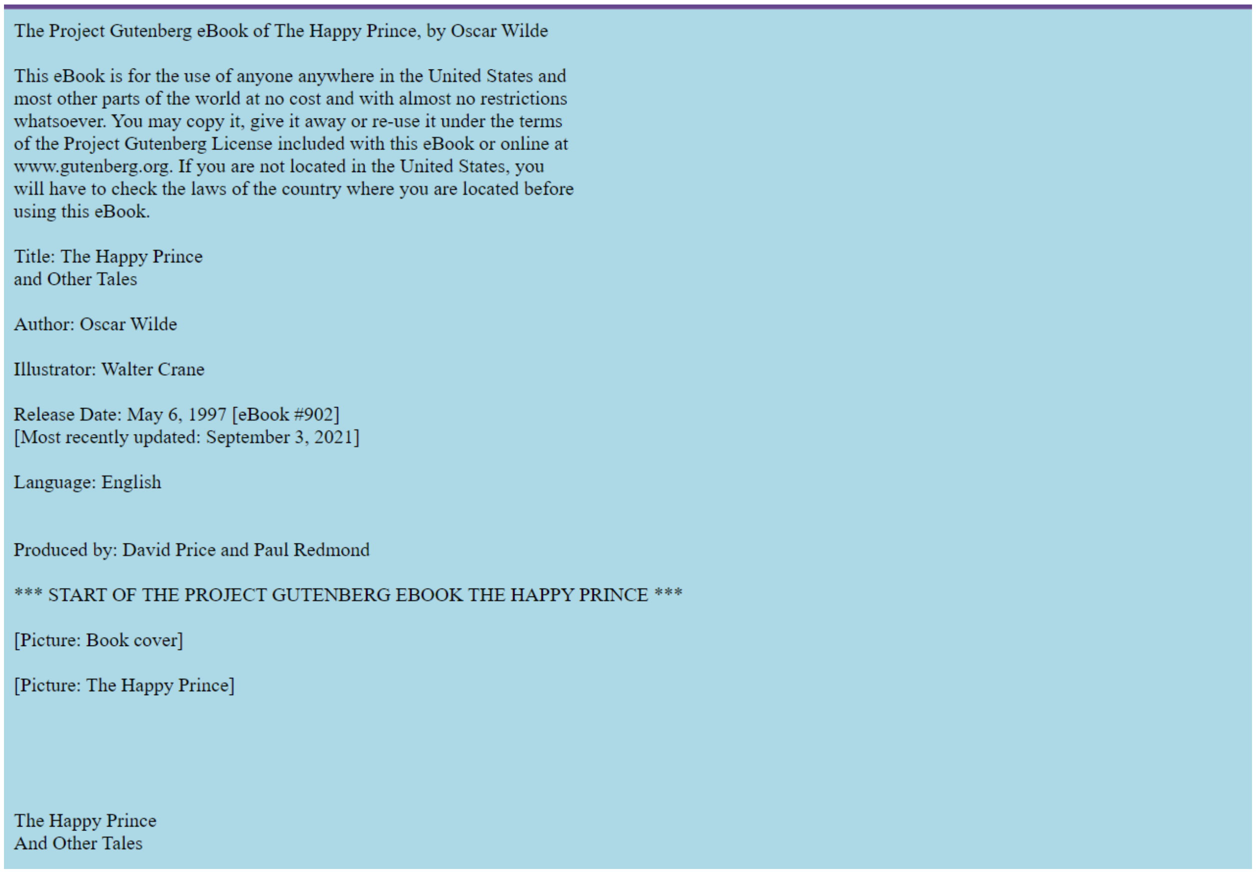
..

divine-comedy.txt

oliver-twist.txt

the-happy-prince.txt

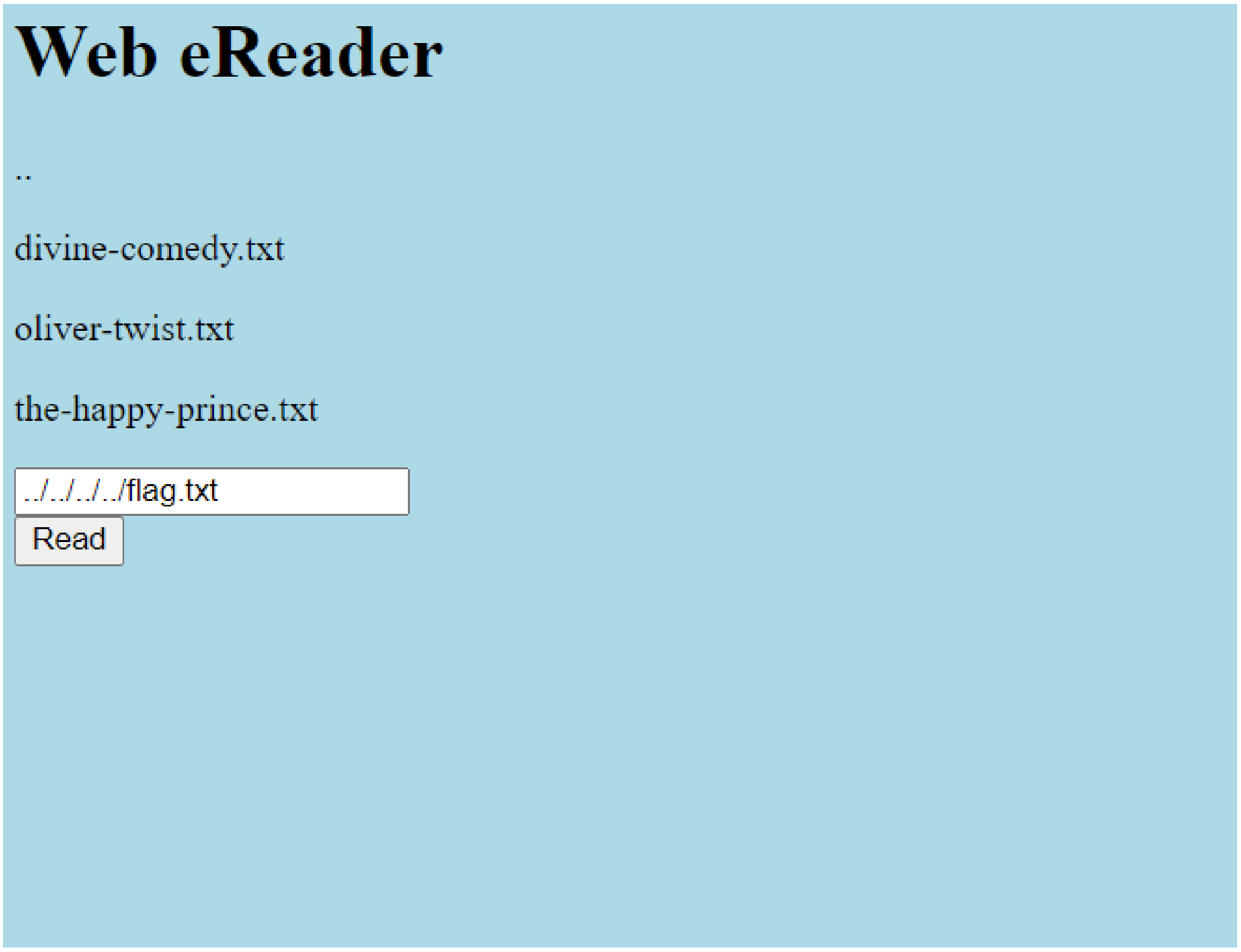
In this website, When we enter the file name from the above list and click on read, the file opens up as shown below for the-happy-prince.txt:



```
The Project Gutenberg eBook of The Happy Prince, by Oscar Wilde

This eBook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this eBook or online at
www.gutenberg.org. If you are not located in the United States, you
will have to check the laws of the country where you are located before
using this eBook.

Title: The Happy Prince
and Other Tales

Author: Oscar Wilde

Illustrator: Walter Crane

Release Date: May 6, 1997 [eBook #902]
[Most recently updated: September 3, 2021]

Language: English


Produced by: David Price and Paul Redmond

*** START OF THE PROJECT GUTENBERG EBOOK THE HAPPY PRINCE ***

[Picture: Book cover]

[Picture: The Happy Prince]




The Happy Prince
And Other Tales
```

And when I tried using /flag.txt in the read box, I was given a “Not Authorized” message.

Then I noticed that I was actually inside 4 directories as shown in the challenge description so I have to climb out of the directories to find the flag. Hence, I executed the following command:



Web eReader

..

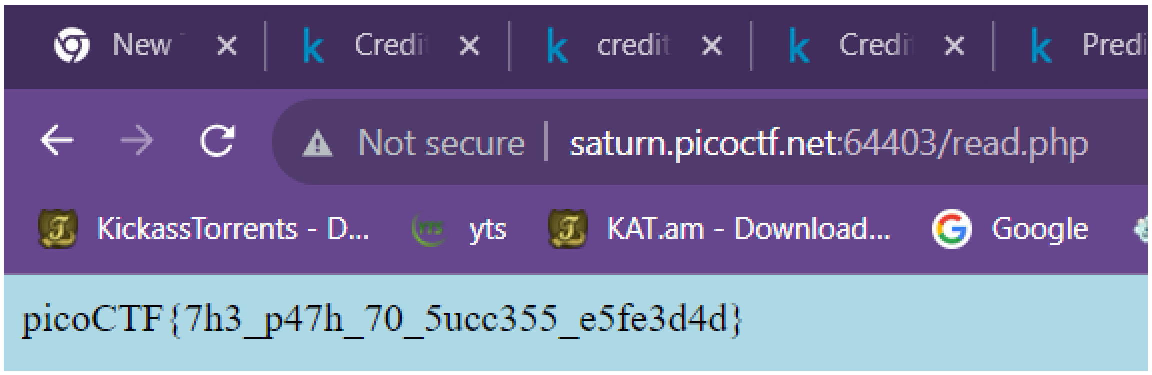
divine-comedy.txt

oliver-twist.txt

the-happy-prince.txt

../../../../flag.txt

Each ../ represents jumping to one parent file hence, to jump four part files, I added this 4 times. After this, when I clicked on search I got the following result:

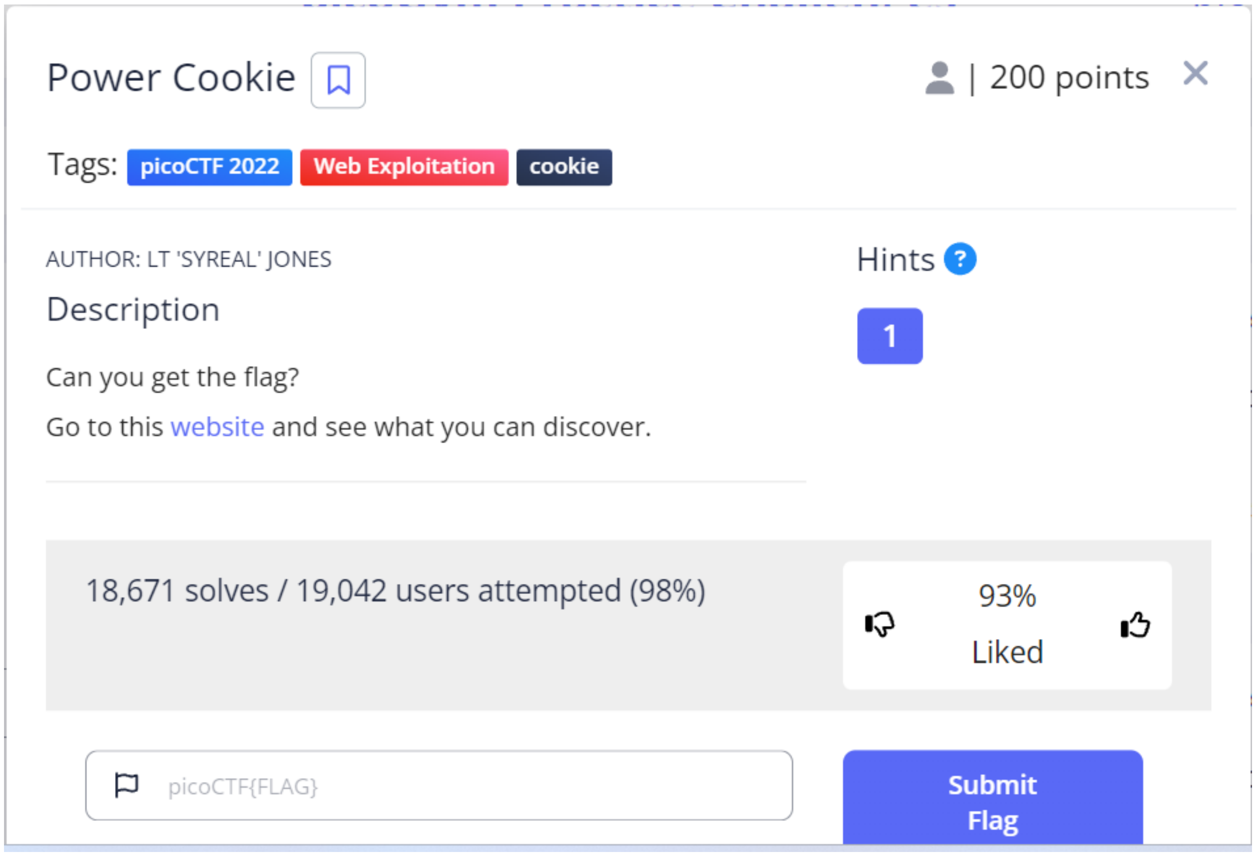


picoCTF{7h3_p47h_70_5ucc355_e5fe3d4d}

As we can see, The flag was successfully captured. Hence, this task was also a success.

Task 7: Power Cookie by picoCTF(200 points):

This challenge is also part of picoCTF and is based on Web Exploitation. The screenshot below gives us a challenge description:



Power Cookie | 200 points

Tags: picoCTF 2022 Web Exploitation cookie

AUTHOR: LT 'SYREAL' JONES

Description

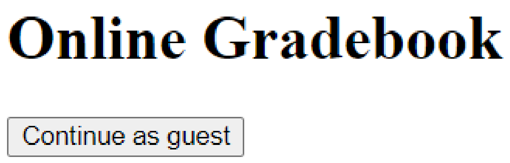
Can you get the flag?

Go to this website and see what you can discover.

18,671 solves / 19,042 users attempted (98%)

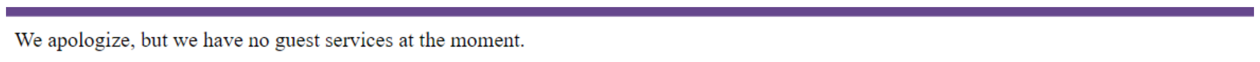
93% Liked

When I clicked on the link and opened the website I got the following result:

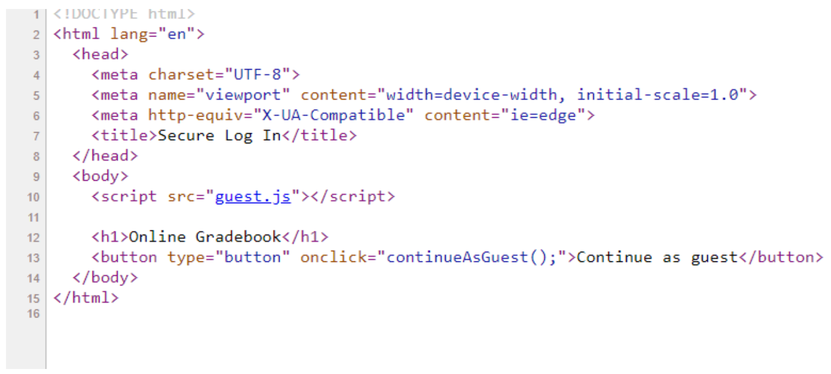


Online Gradebook

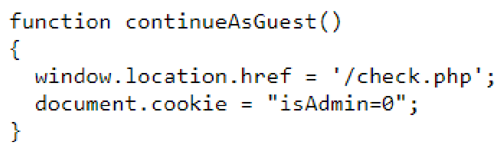
Next, I clicked on Continue as guest and got the following:


We apologize, but we have no guest services at the moment.

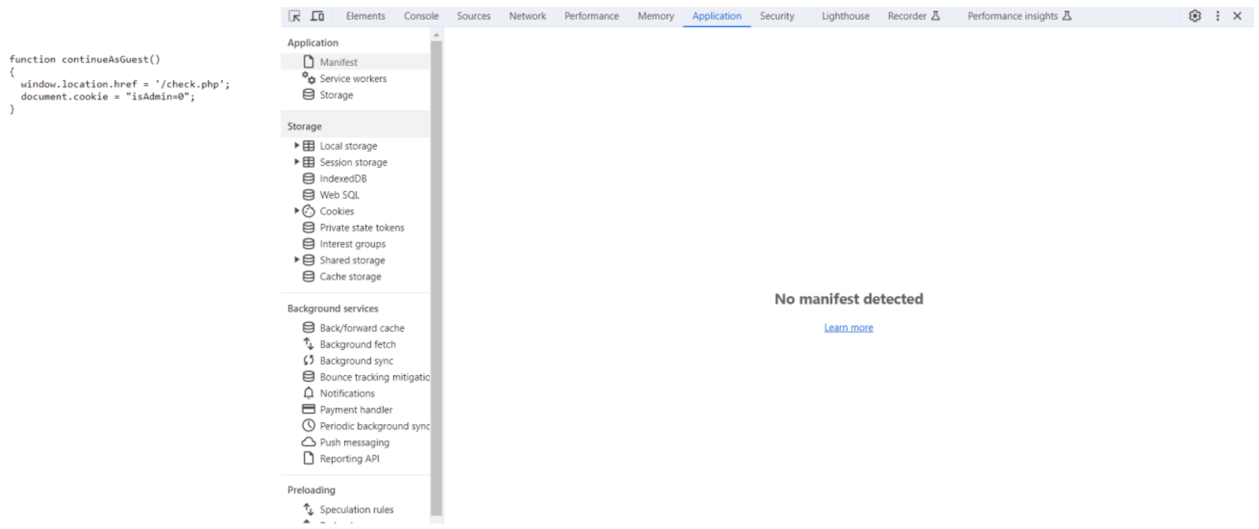
Next step as always was to inspect the wepage:


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Secure Log In</title>
  </head>
  <body>
    <script src="guest.js"></script>

    <h1>Online Gradebook</h1>
    <button type="button" onclick="continueAsGuest();">Continue as guest</button>
  </body>
</html>
```

When I clicked on guest.js, I got the following:

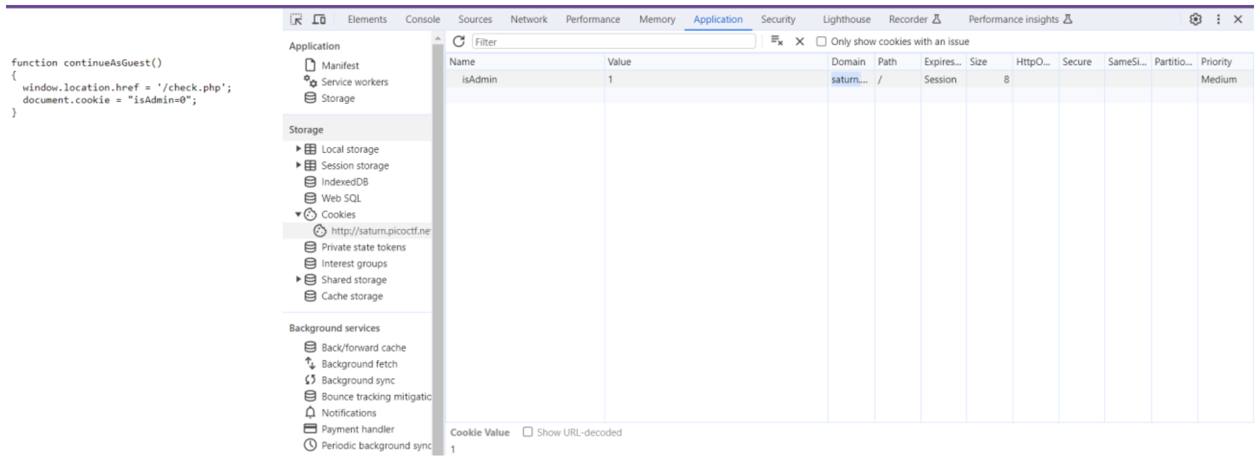

```
function continueAsGuest()
{
  window.location.href = '/check.php';
  document.cookie = "isAdmin=0";
}
```

Next, I opened up developer tools and went to application:



```
function continueAsGuest()
{
  window.location.href = '/check.php';
  document.cookie = "isAdmin=0";
}
```

Then I went to the cookie option and changed the value from 0 to 1 as shown below:

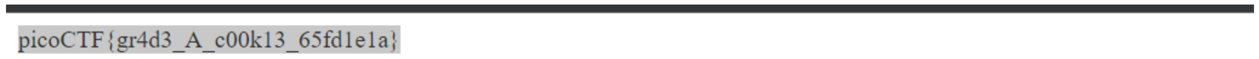


```
function continueAsGuest()
{
  window.location.href = '/check.php';
  document.cookie = "isAdmin=0";
}
```

Name	Value	Domain	Path	Expires...	Size	HttpO...	Secure	SameSi...	Partitio...	Priority
isAdmin	1	saturn....	/	Session	8					Medium

Cookie Value ☐ Show URL-decoded
1

I didn’t have to change any other values for this task. After this I hard refreshed the webpage and found the following result:

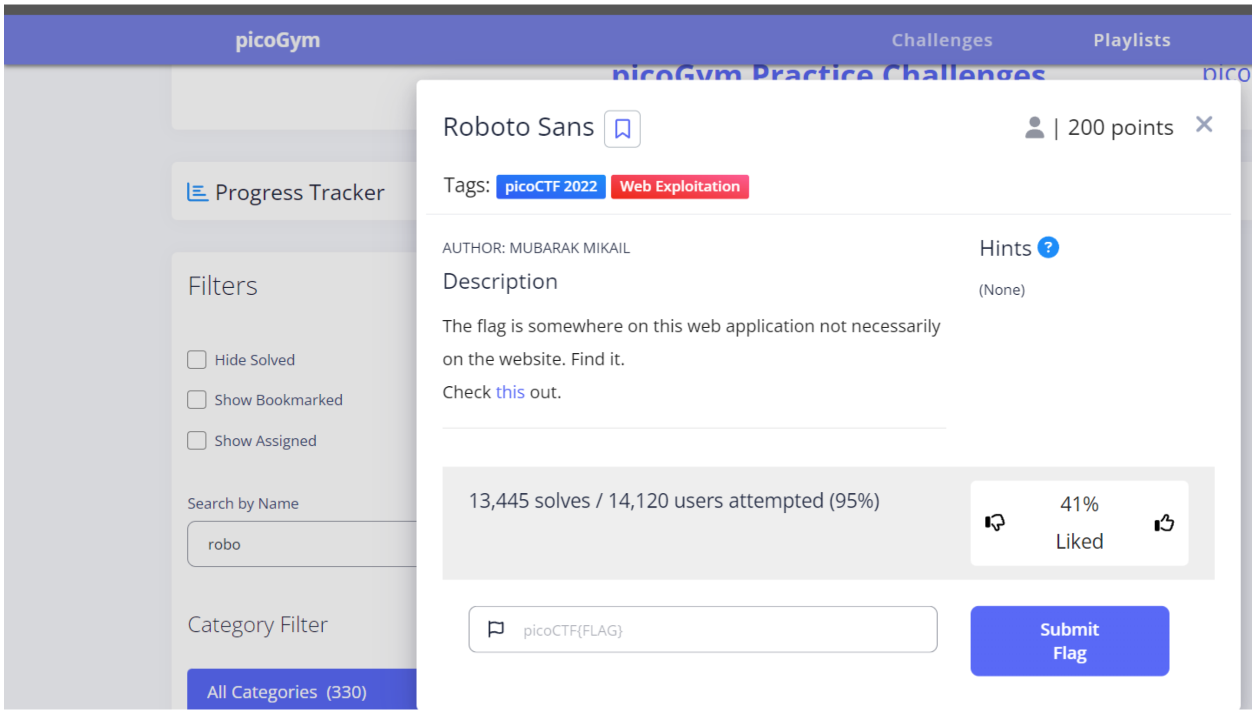


picoCTF{gr4d3_A_c00k13_65fd1e1a}

Hence, I was successful able to capture the flag for this challenge as well.

Task 8: Roboto Sans by picoCTF(200 points):

This challenge is part of picoCTF 2021. The screenshot below gives a description of the challenge:



Roboto Sans | 200 points

Tags: picoCTF 2022 Web Exploitation

AUTHOR: MUBARAK MIKAIL

Description

The flag is somewhere on this web application not necessarily on the website. Find it.

Check this out.

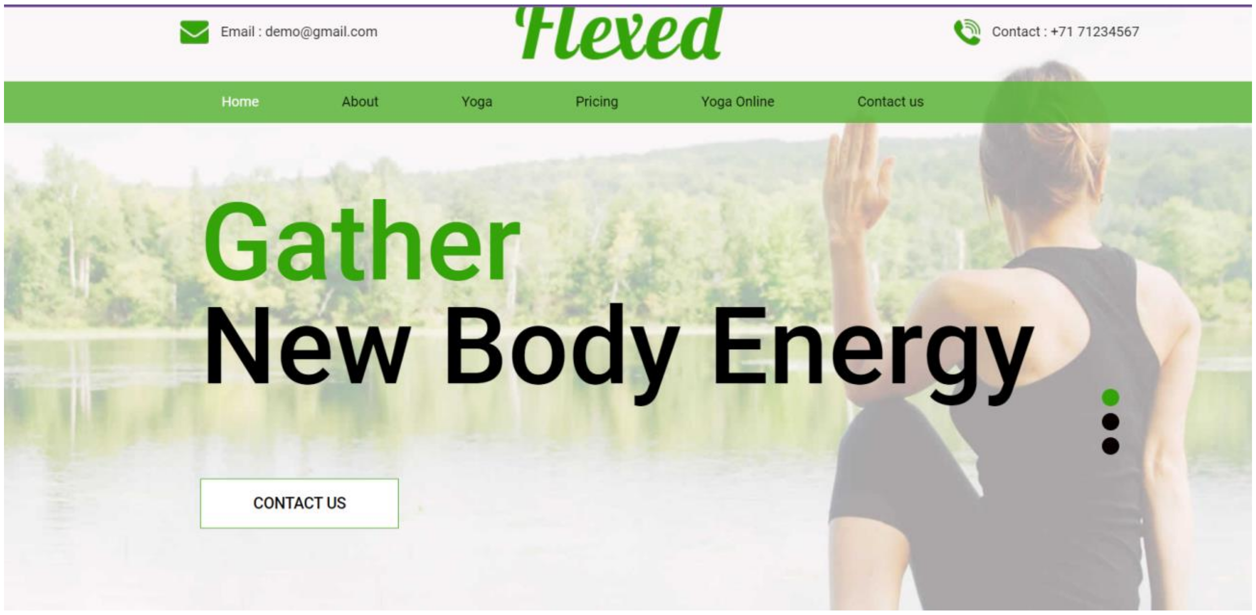
Hints

(None)

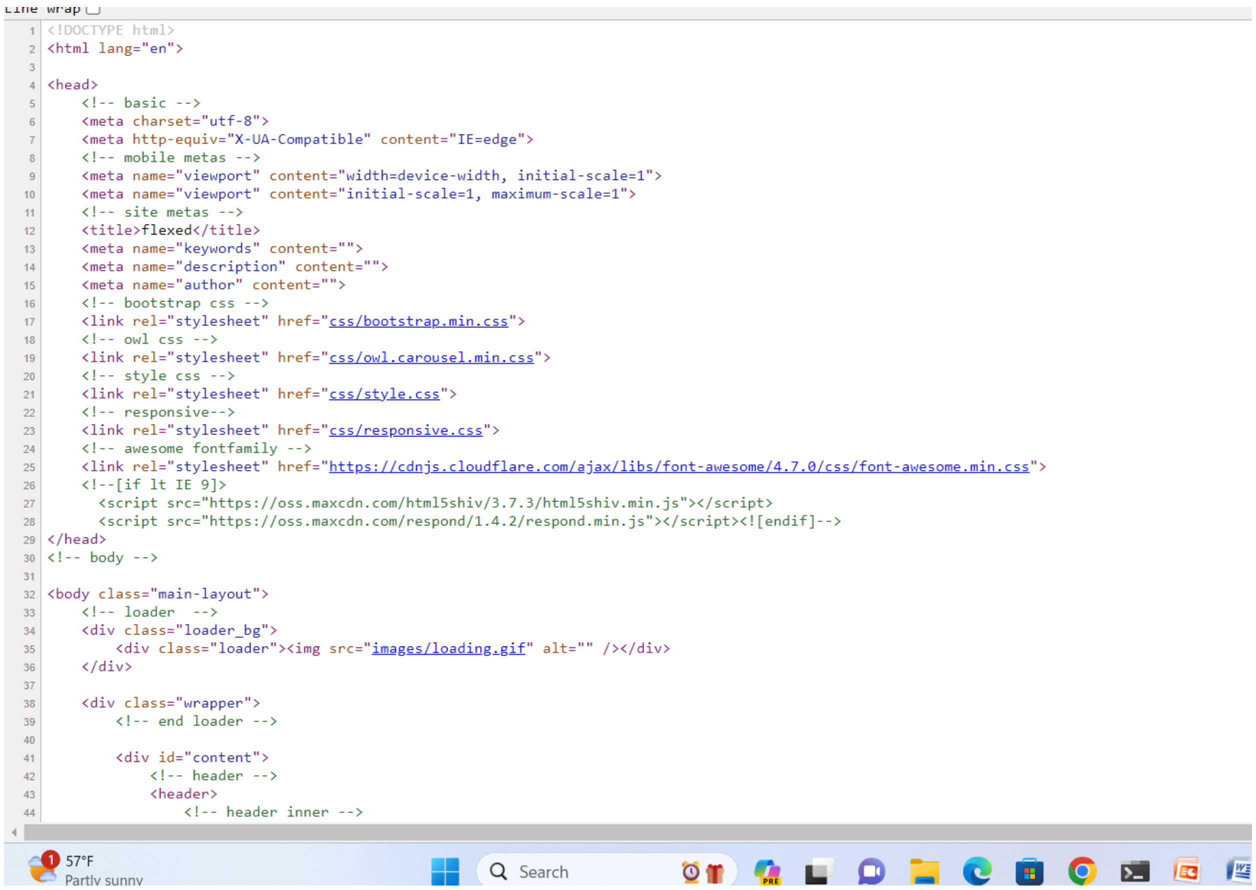
13,445 solves / 14,120 users attempted (95%)

41% Liked

After clicking the link, The following website opened up:

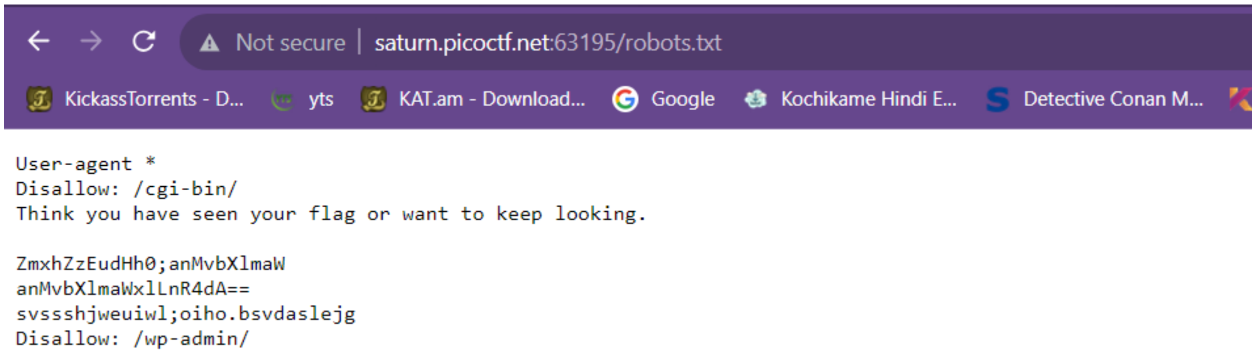


As we can see, it is the same webpage as a previous challenge “search source”. Next, I went to the inspect and saw the following code:



```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- basic -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- mobile metas -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <!-- site metas -->
    <title>flexed</title>
    <meta name="keywords" content="">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- bootstrap css -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <!-- owl css -->
    <link rel="stylesheet" href="css/owl.carousel.min.css">
    <!-- style css -->
    <link rel="stylesheet" href="css/style.css">
    <!-- responsive-->
    <link rel="stylesheet" href="css/responsive.css">
    <!-- awesome fontfamily -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <!--[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]-->
</head>
<!-- body -->

<body class="main-layout">
    <!-- loader  -->
    <div class="loader_bg">
        <div class="loader"><img src="images/loading.gif" alt="" /></div>
    </div>

    <div class="wrapper">
        <!-- end loader -->

        <div id="content">
            <!-- header -->
            <header>
                <!-- header inner -->
```

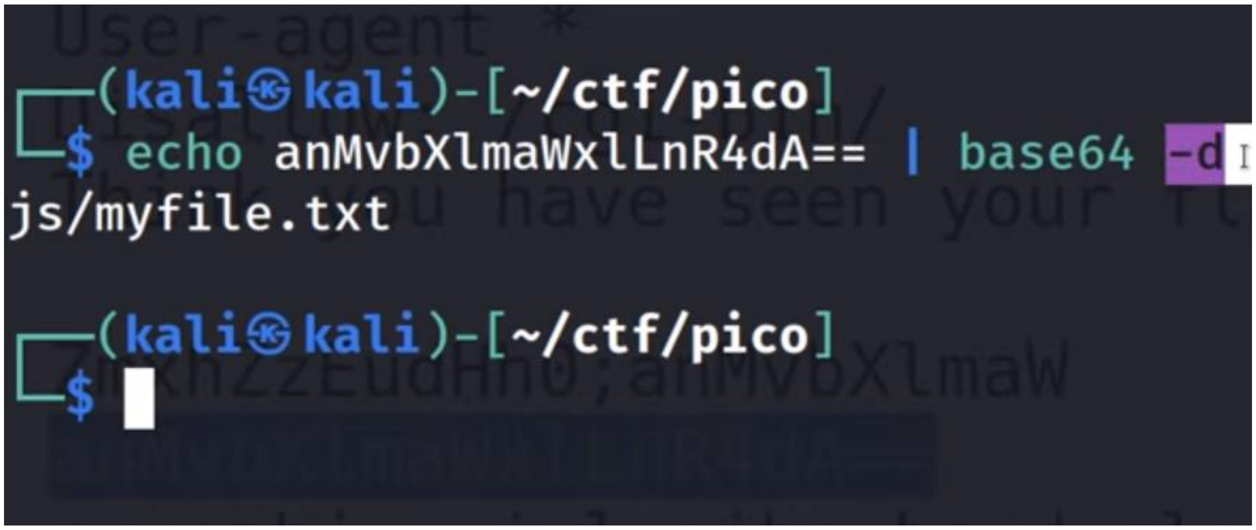
Next, I looked at the developer tools as well but the flag was nowhere to be found. Finally, I checked on the link if a robots.txt even exists or not:



Not secure | saturn.picoctf.net:63195/robots.txt

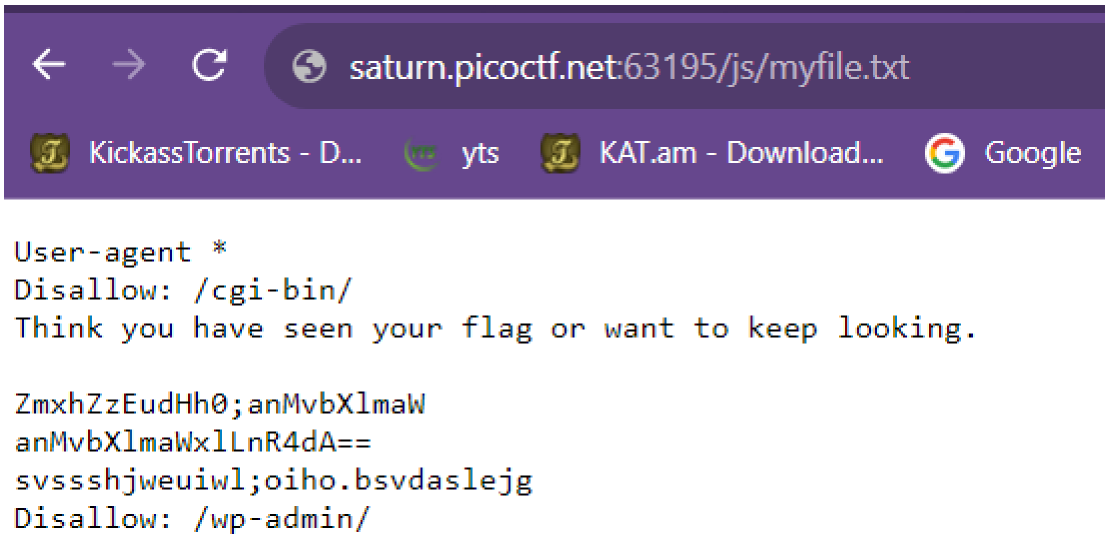
```
User-agent *
Disallow: /cgi-bin/
Think you have seen your flag or want to keep looking.

ZmxhZzEudHh0;anMvbXlmaW
anMvbXlmaWxlLnR4dA==
svssshjweuiwl;oiho.bsvdaslejg
Disallow: /wp-admin/
```

The screenshot above confirms that a file named robots.txt does exist. I used echo for the given command and saw the following result:

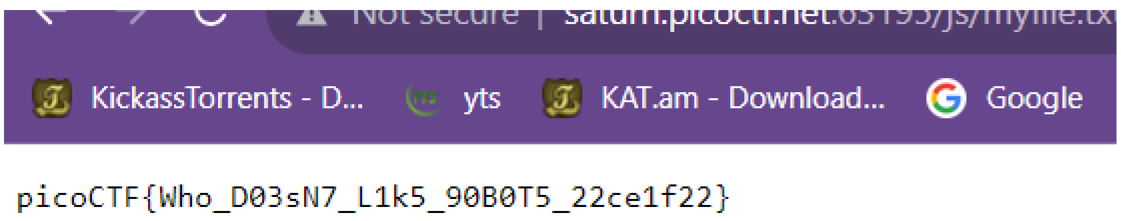


```
┌──(kali㉿kali)-[~/ctf/pico]
└─$ echo anMvbXlmaWxlLnR4dA== | base64 -d
js/myfile.txt

┌──(kali㉿kali)-[~/ctf/pico]
└─$ 
```

It is pointing us towards a js/myfile.txt. Then, I used this newfound knowledge to access the file using the link as shown below:



```
User-agent *
Disallow: /cgi-bin/
Think you have seen your flag or want to keep looking.

ZmxhZzEudHh0;anMvbXlmaW
anMvbXlmaWxlLnR4dA==
svssshjweuiwl;oiho.bsvdaslejg
Disallow: /wp-admin/
```

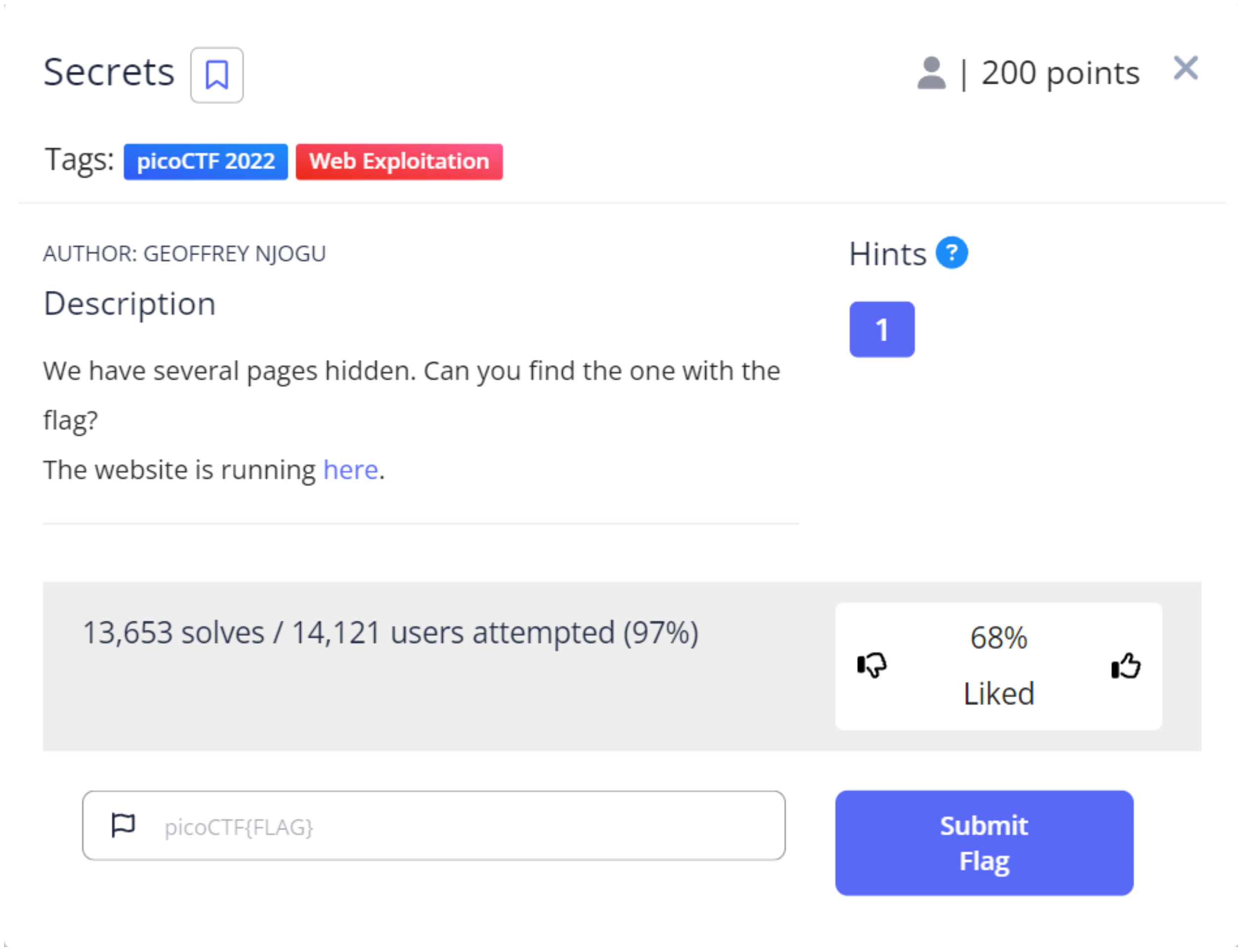


```
picoCTF{Who_D03sN7_L1k5_90B0T5_22ce1f22}
```

Hence, I successfully captured the flag for this challenge as well.

Task 9: Secrets by picoCTF(200 points):

This task is part of the picoCTF 2021 challenge. The screenshot below gives us the challenge description:


Secrets

| 200 points

Tags: picoCTF 2022 Web Exploitation

AUTHOR: GEOFFREY NJOGU

Description

We have several pages hidden. Can you find the one with the flag?

The website is running here.

Hints

1

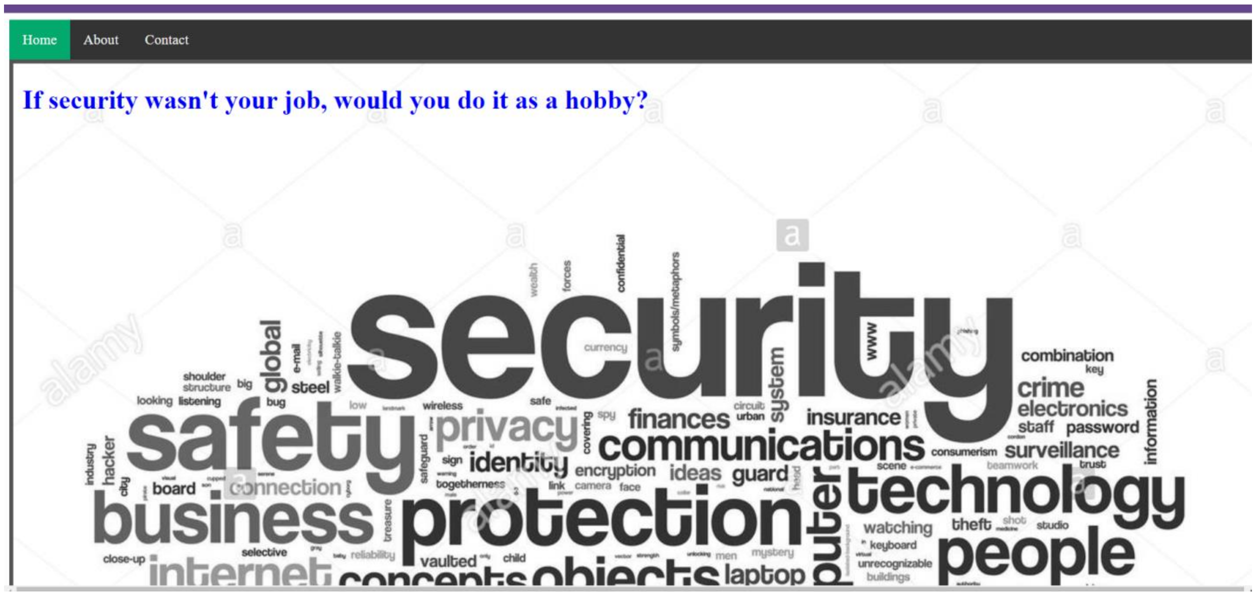
13,653 solves / 14,121 users attempted (97%)

68% Liked

picoCTF{FLAG}

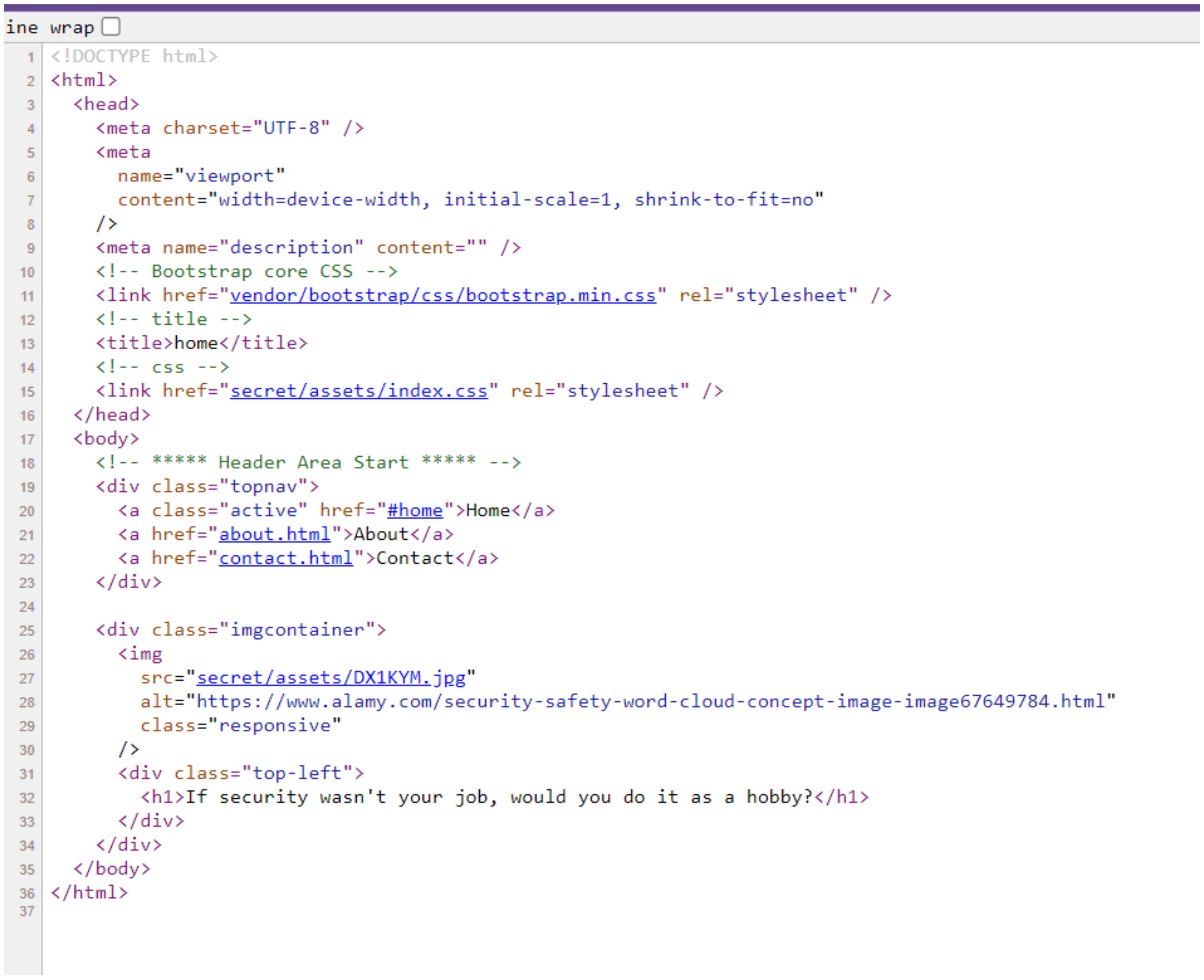
Submit Flag

When I clicked on the given link, I got the following:



If security wasn't your job, would you do it as a hobby?

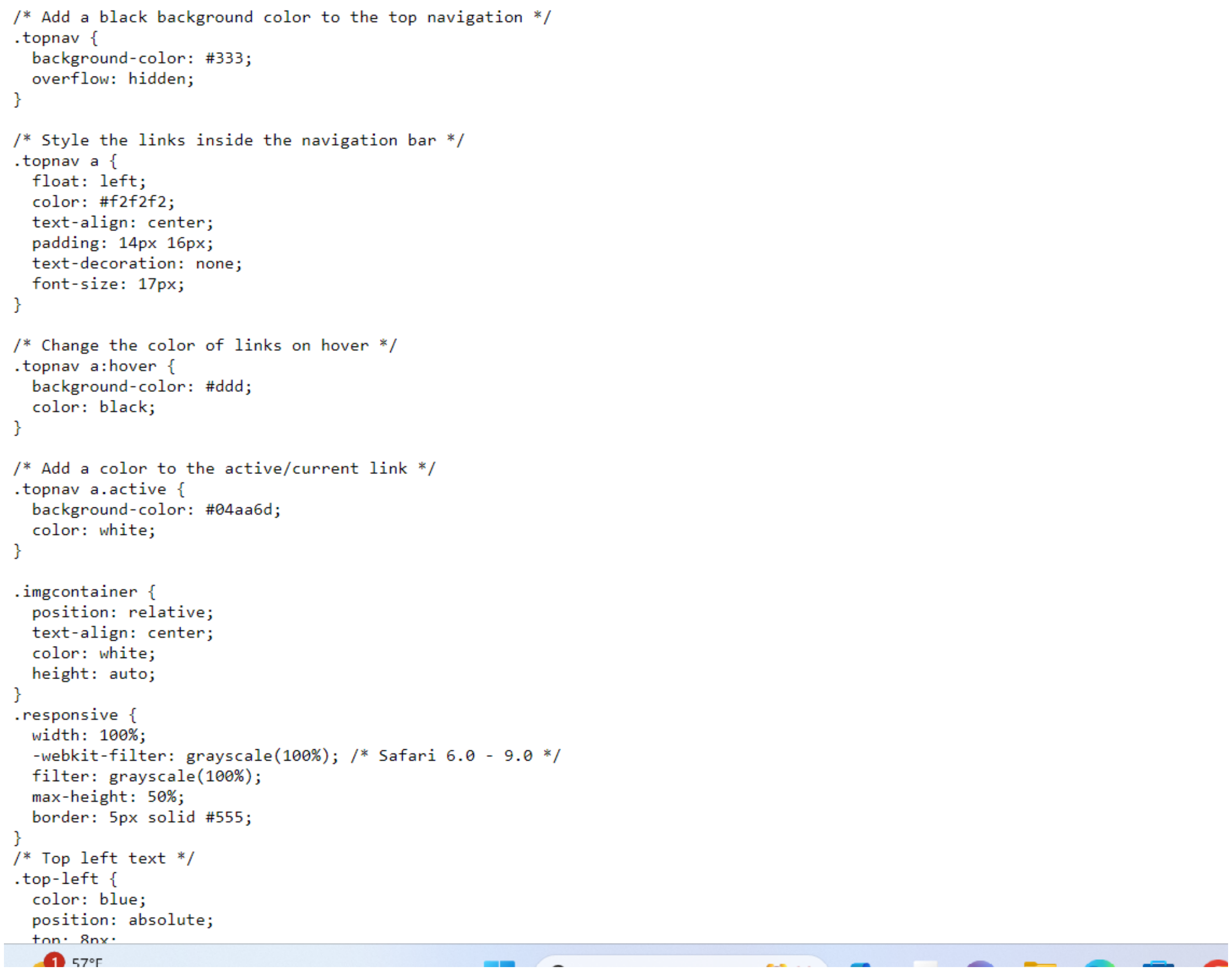
As the first step, I opened the source code of the webpage and found the following:



ine wrap

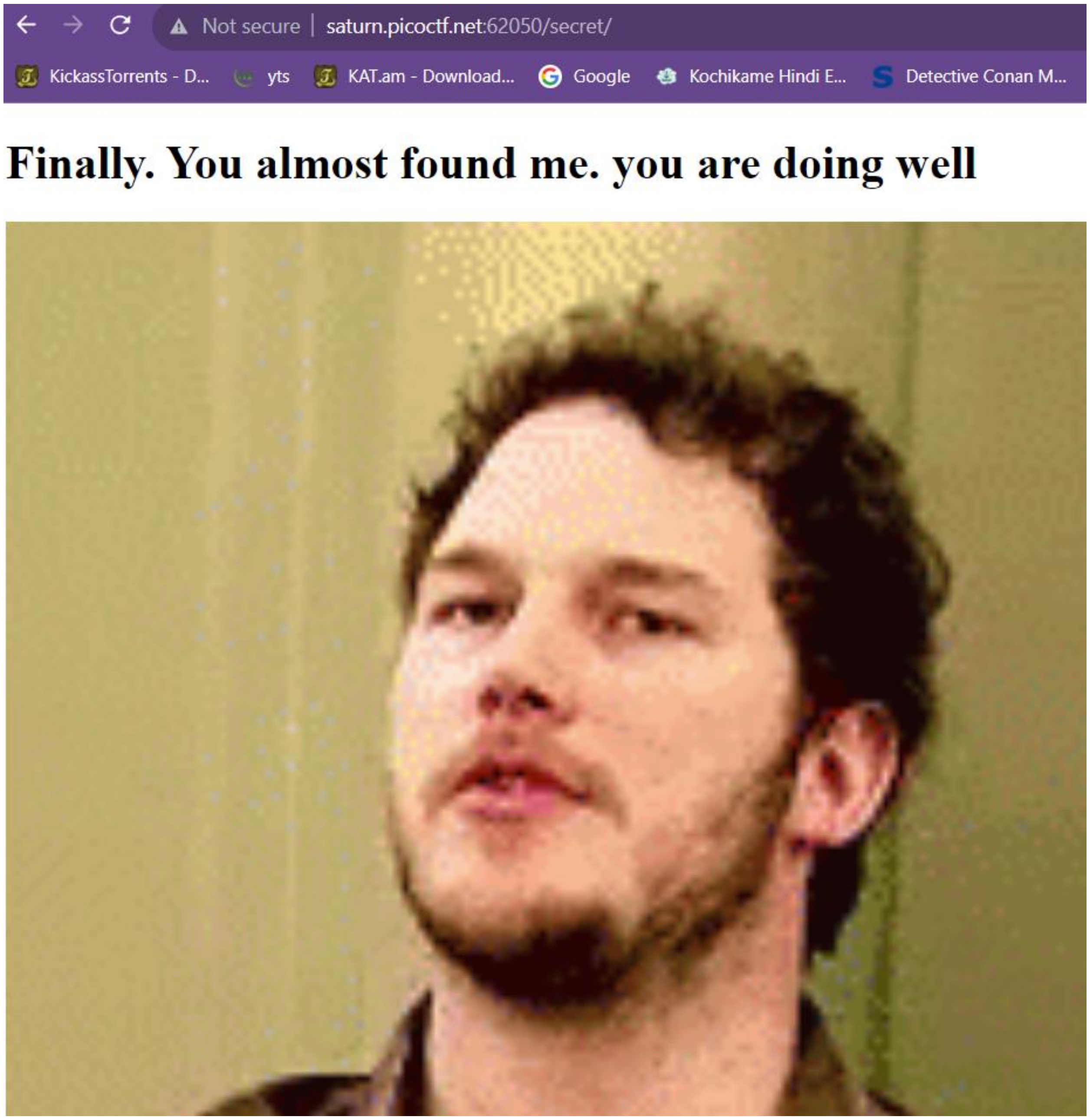
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />
    <meta name="description" content="" />
    <!-- Bootstrap core CSS -->
    <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
    <!-- title -->
    <title>home</title>
    <!-- css -->
    <link href="secret/assets/index.css" rel="stylesheet" />
  </head>
  <body>
    <!-- ***** Header Area Start ***** -->
    <div class="topnav">
      <a class="active" href="#home">Home</a>
      <a href="about.html">About</a>
      <a href="contact.html">Contact</a>
    </div>

    <div class="imgcontainer">
      <img
        src="secret/assets/DX1KYM.jpg"
        alt="https://www.alamy.com/security-safety-word-cloud-concept-image-image67649784.html"
        class="responsive"
      />
      <div class="top-left">
        <h1>If security wasn't your job, would you do it as a hobby?</h1>
      </div>
    </div>
  </body>
</html>
```

Next, I opened the index.css file and found the following result:



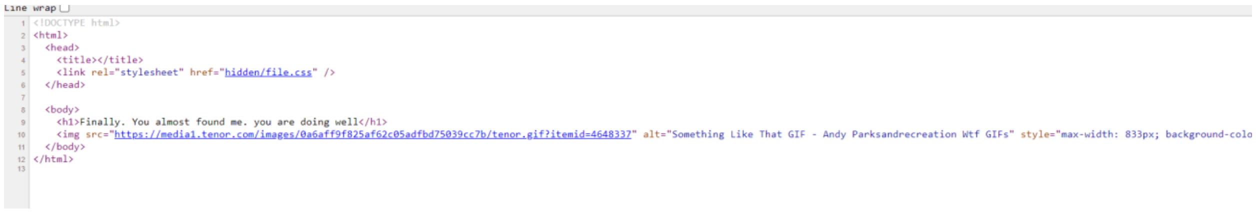
```
/* Add a black background color to the top navigation */
.topnav {
  background-color: #333;
  overflow: hidden;
}

/* Style the links inside the navigation bar */
.topnav a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

/* Change the color of links on hover */
.topnav a:hover {
  background-color: #ddd;
  color: black;
}

/* Add a color to the active/current link */
.topnav a.active {
  background-color: #04aa6d;
  color: white;
}

.imgcontainer {
  position: relative;
  text-align: center;
  color: white;
  height: auto;
}
.responsive {
  width: 100%;
  -webkit-filter: grayscale(100%); /* Safari 6.0 - 9.0 */
  filter: grayscale(100%);
  max-height: 50%;
  border: 5px solid #555;
}
/* Top left text */
.top-left {
  color: blue;
  position: absolute;
  top: 8px;
```

I couldn’t find the flag here. The next step was to check the weblink, I edited the weblink to go back to the folder and saw the following:



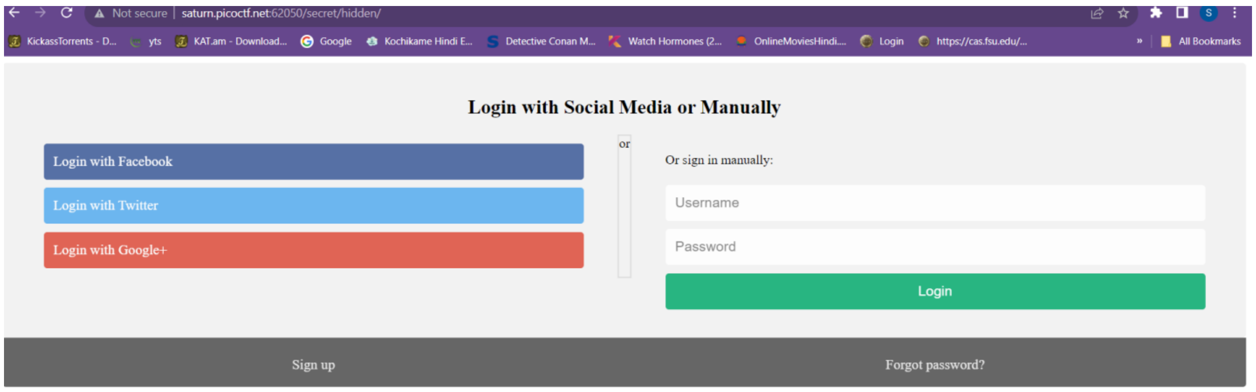
Finally. You almost found me. you are doing well

Next, I opened the source code this new webpage that I found and saw the following:

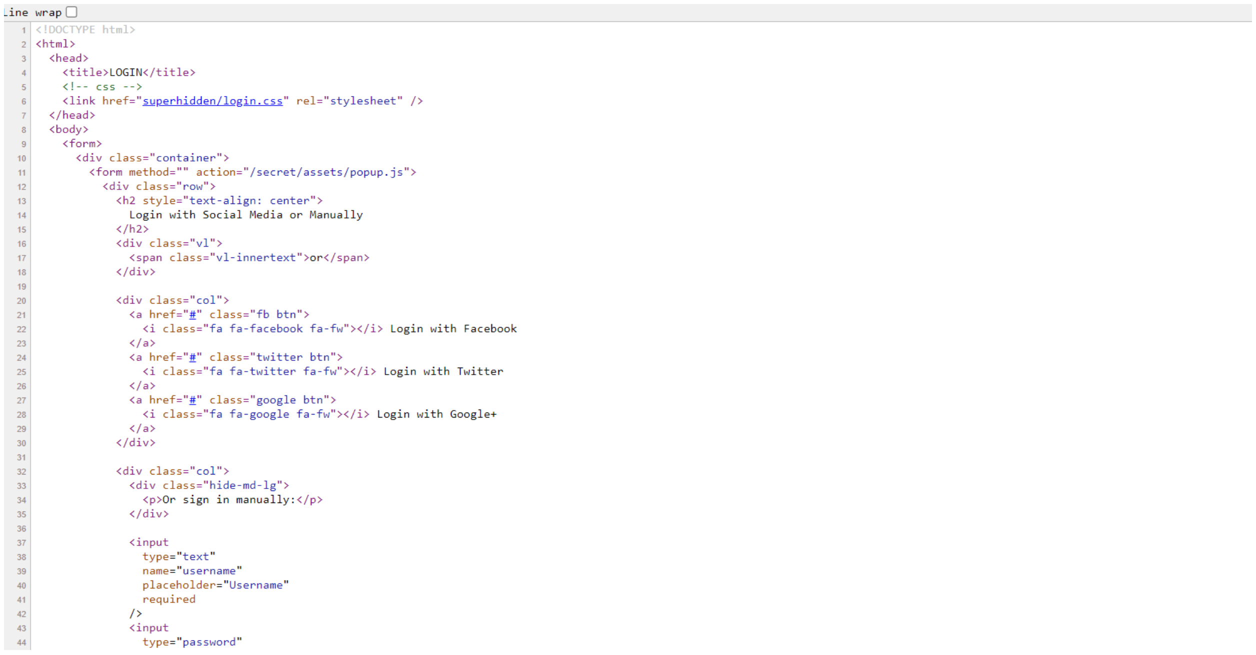


```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <link rel="stylesheet" href="hidden/file.css" />
  </head>

  <body>
    <h1>Finally. You almost found me. you are doing well</h1>
    <img src="https://media1.tenor.com/images/0a6aff9f825af62c05adfbd75039cc7b/tenor.gif?itemid=4648337" alt="Something Like That GIF - Andy Parksandrecreation Wtf GIFs" style="max-width: 833px; background-colo
  </body>
</html>
```

After this, I went to the web link again and tried to jump back to the folders and found the following:

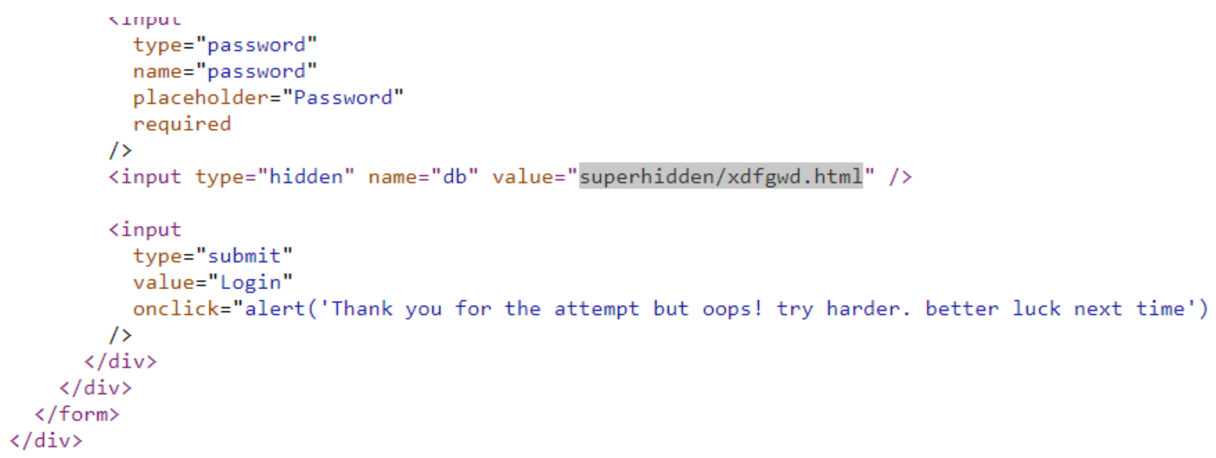


I just removed file.css from the end of the link to get the above result. After this, I went to this new webpage source code and got the following:

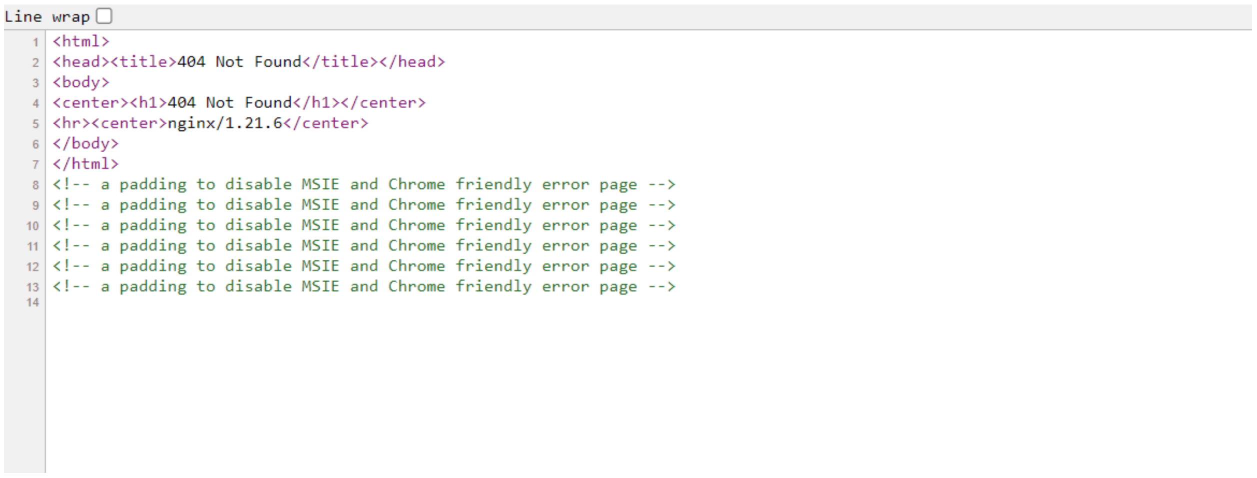


```
<!DOCTYPE html>
<html>
  <head>
    <title>LOGIN</title>
    <!-- css -->
    <link href="superhidden/login.css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <div class="container">
        <form method="" action="/secret/assets/popup.js">
          <div class="row">
            <h2 style="text-align: center">
              Login with Social Media or Manually
            </h2>
            <div class="vl">
              <span class="vl-innertext">or</span>
            </div>

            <div class="col">
              <a href="#" class="fb btn">
                <i class="fa fa-facebook fa-fw"></i> Login with Facebook
              </a>
              <a href="#" class="twitter btn">
                <i class="fa fa-twitter fa-fw"></i> Login with Twitter
              </a>
              <a href="#" class="google btn">
                <i class="fa fa-google fa-fw"></i> Login with Google+
              </a>
            </div>

            <div class="col">
              <div class="hide-md-lg">
                <p>Or sign in manually:</p>
              </div>

              <input
                type="text"
                name="username"
                placeholder="Username"
                required
              />
              <input
                type="password"
```

I observed a hiddentext html link in this source code. I went ahead and added that link in the web link and got the following:



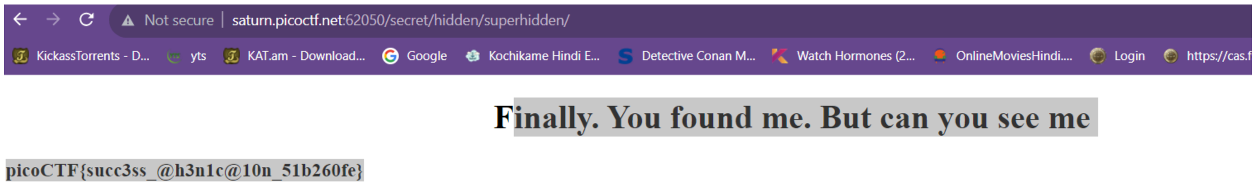
```
          <input
            type="password"
            name="password"
            placeholder="Password"
            required
          />
          <input type="hidden" name="db" value="superhidden/xdfgwd.html" />

          <input
            type="submit"
            value="Login"
            onclick="alert('Thank you for the attempt but oops! try harder. better luck next time')
          />
        </div>
      </div>
    </form>
  </div>
```



Line wrap

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.21.6</center>
</body>
</html>
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
```

Finally, I edited the web link again according to my need to jump back files and found the following:



Not secure | saturn.picoctf.net:62050/secret/hidden/superhidden/

KickassTorrents - D... yts KAT.am - Download... Google Kochikame Hindi E... Detective Conan M... Watch Hormones (2... OnlineMoviesHindi.... Login https://cas.f

Finally. You found me. But can you see me

picoCTF{succ3ss_@h3n1c@10n_51b260fe}

As we can see, the flag was hidden in the same ink color as background here. Hence, the flag was successfully captured.

……

I have completed challenges worth 1300 points in the picoCTF website which is equal to 130 points in the assignments. This includes 100 points for the challenge and 300 points for bonus. My username for these challenges was sk22bo and my name can be successfully found on the leader board at the picoCTF website.

For any further verification the username and password for the picoCTF challenge login is given below:

Username- sk22bo

Password- Offensive123

THE END