CTF Report

Full Name: Chirag Suthar
Program: HCS - Penetration Testing 1-Month Internship
Date: 11/03/2025

Category: Web 2.0

Sub-Category: Look Web

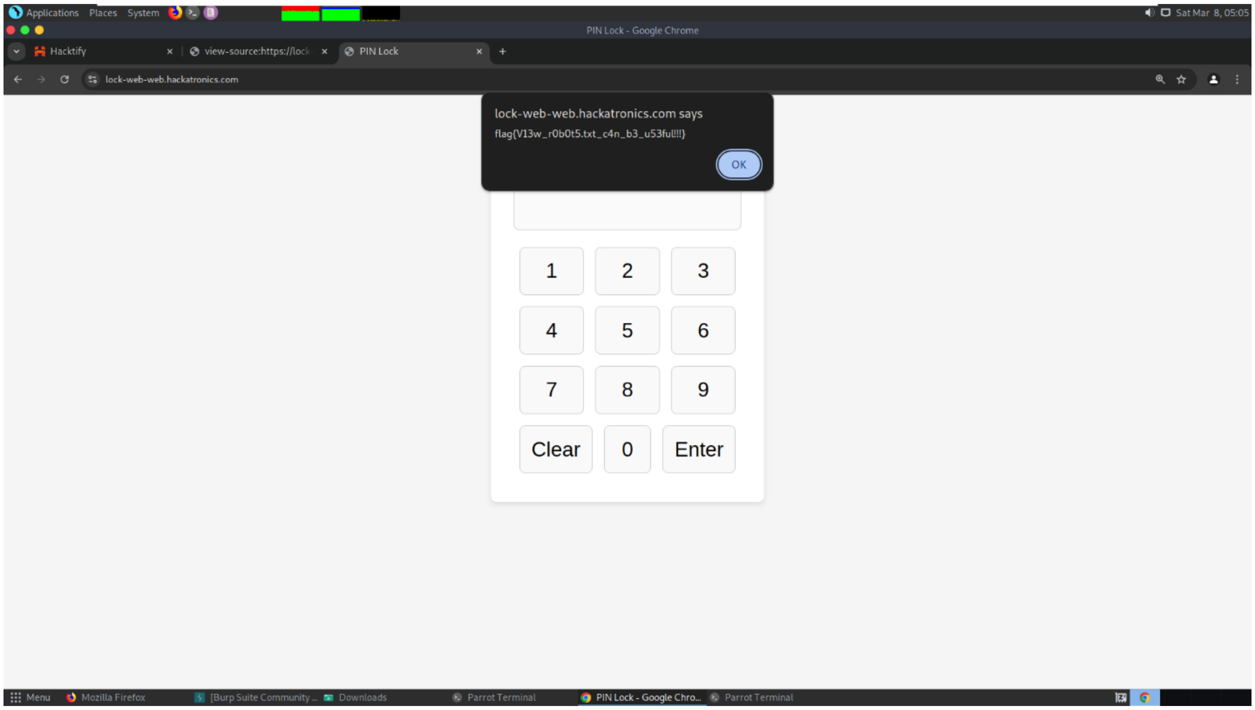
Description: The *Web* 2.0 CTF challenge involved exploring web application security concepts, requiring critical thinking and problem-solving to uncover hidden flaws and achieve the intended objectives.

Challenge Overview: The *Look Web* CTF challenge required exploring the web application's structure. The flag was obtained by appending /robots.txt to the URL, revealing hidden information that led to the solution.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Performed basic analysis of the website's structure, identifying accessible endpoints and observing the website's behavior.
2. **Input Validation Testing:** Tested various inputs in URL parameters and forms to check for unusual responses or hidden content.
3. **Directory Enumeration:** Used common directory enumeration techniques and tools to identify /robots.txt as a potential entry point.
4. **Exploitation:** Accessed /robots.txt, which revealed disallowed paths containing critical information.
5. **Flag Retrieval:** Navigated to the disclosed path from /robots.txt and successfully obtained the flag.

Flag: flag{V13w_r0b0t5.txt_c4n_b3_u53ful!!!}



lock-web-web.hackatronics.com says

flag{V13w_r0b0t5.txt_c4n_b3_u53ful!!!}

Category: Web 2.0

Sub-Category: The World

Description: The *Web 2.0* CTF challenge involved exploring web application security concepts, requiring critical thinking and problem-solving to uncover hidden flaws and achieve the intended objectives.

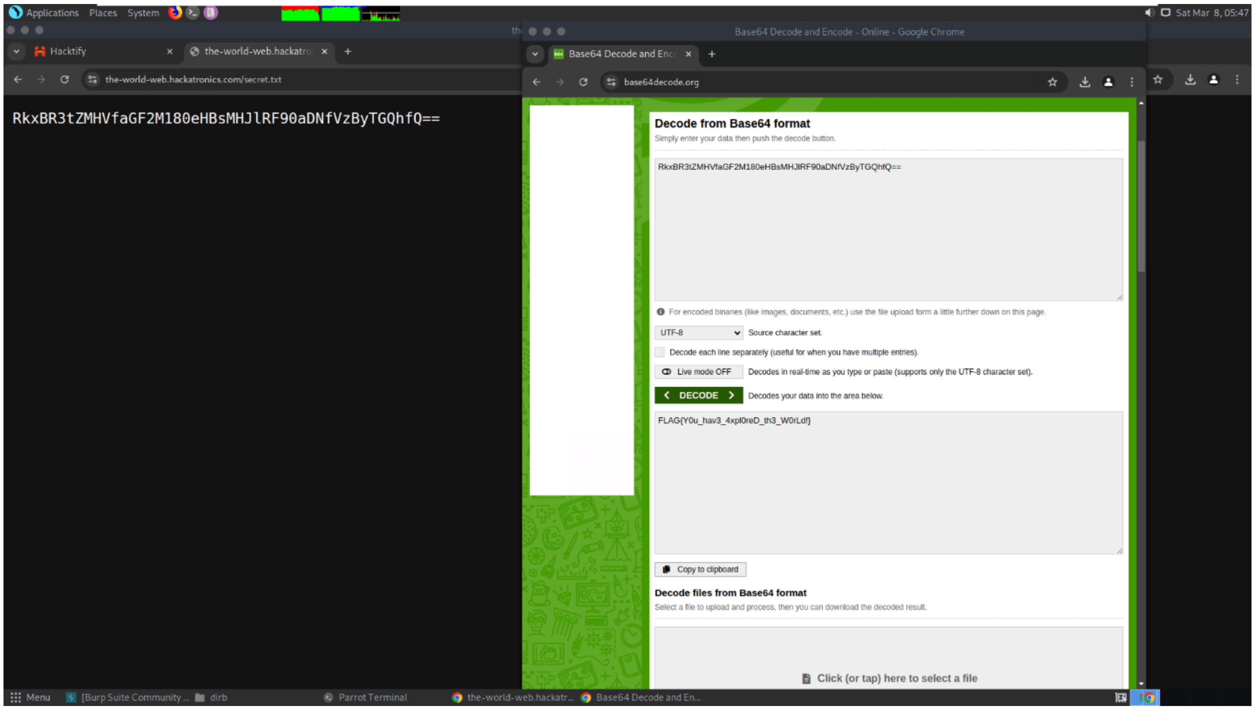
Challenge Overview: This challenge required identifying hidden paths within the website. By appending /secret.txt to the URL, a concealed file was discovered, ultimately revealing the flag.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Conducted a thorough analysis of the website's structure, URLs, and page behavior to identify potential entry points.
2. **Input Validation Testing:** Tested various inputs in URL parameters and forms to check for unusual responses or hidden content.
3. **Directory Enumeration:** Performed directory brute-forcing using tools like **gobuster** or **dirbuster**, revealing accessible endpoints.
4. **Exploitation:** Accessed /robots.txt, which revealed disallowed paths containing critical information in the encoded format.
5. **Flag Retrieval:** Navigated to the disclosed path from /robots.txt and successfully obtained the flag, followed by decoding the obtained string using base64 decoder tool.

Encoded String: RkxBR3tZMHVfaGF2M180eHBsMHJlRF90aDNfVzByTGQhfQ==

Flag: FLAG{Y0u_hav3_4xpl0reD_th3_W0rLd!}



the-world-web.hackatronics.com/secret.txt

```
RkxBR3tZMHVfaGF2M180eHBsMHJlRF90aDNfVzByTGQhfQ==
```

Base64 Decode and Encode - Online - Google Chrome

base64decode.org

Decode from Base64 format

Simply enter your data then push the decode button.

```
RkxBR3tZMHVfaGF2M180eHBsMHJlRF90aDNfVzByTGQhfQ==
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

□ Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

```
FLAG{Y0u_hav3_4xpl0reD_th3_W0rLd!}
```

Category: Network Forensics

Sub-Category: Corrupted

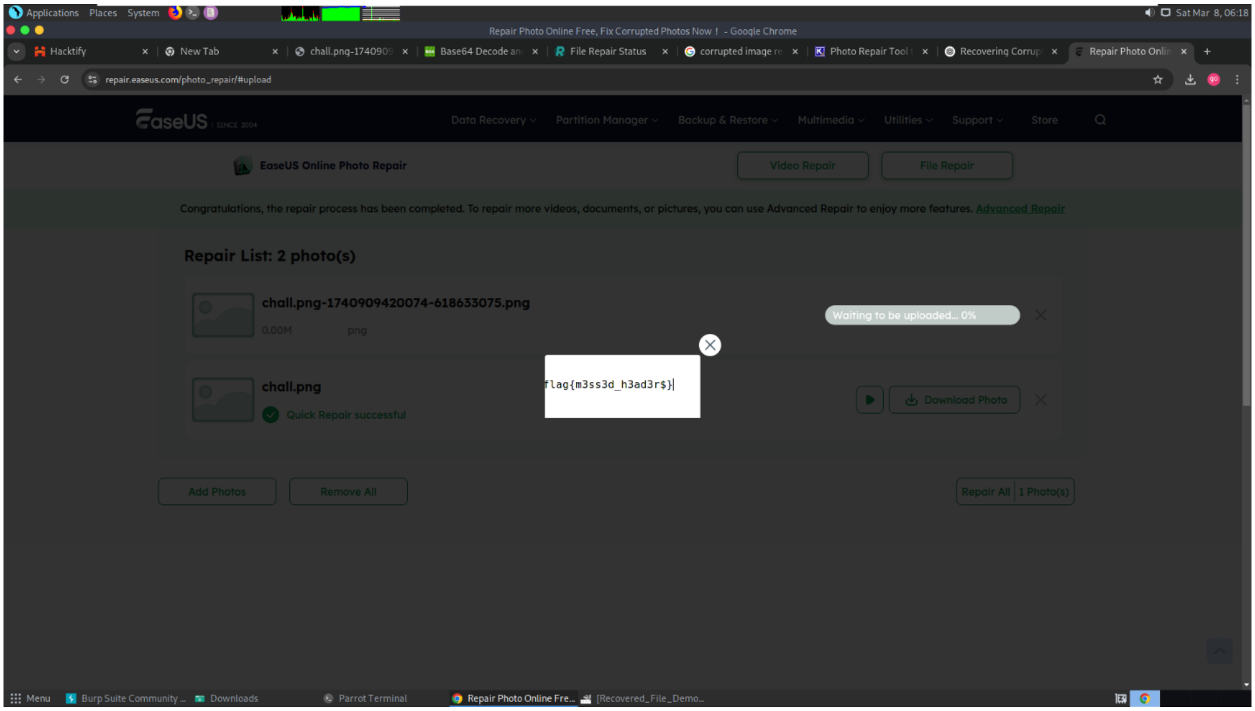
Description: The Network Forensics challenge involved analyzing captured network traffic to uncover hidden data. By carefully examining packet details, patterns, and communication flows, the investigation led to the successful extraction of the concealed flag.

Challenge Overview: The challenge involved analyzing a corrupted image file. Using the 'repair.easeus.com' portal, the image was successfully repaired, revealing the hidden flag text embedded within the recovered photo.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Examined the provided image file and identified that it was corrupted or unreadable.
2. **Input Validation Testing:** Attempted to open the image in multiple viewers to confirm corruption and verify file integrity.
3. **Directory Enumeration:** Explored potential metadata or file details but found no immediate clues.
4. **Exploitation:** Utilized the 'repair.easeus.com' portal to repair the corrupted image file.
5. **Flag Retrieval:** Successfully recovered the image, which revealed the hidden flag text within the restored content.

Flag: flag{m3ss3d_h3ad3r$}



Congratulations, the repair process has been completed. To repair more videos, documents, or pictures, you can use Advanced Repair to enjoy more features. Advanced Repair

Repair List: 2 photo(s)

chall.png-1740909420074-618633075.png
0.00M png
Waiting to be uploaded... 0%

chall.png
Quick Repair successful

```
flag{m3ss3d_h3ad3r$}
```

Category: Network Forensics

Sub-Category: Shadow Web

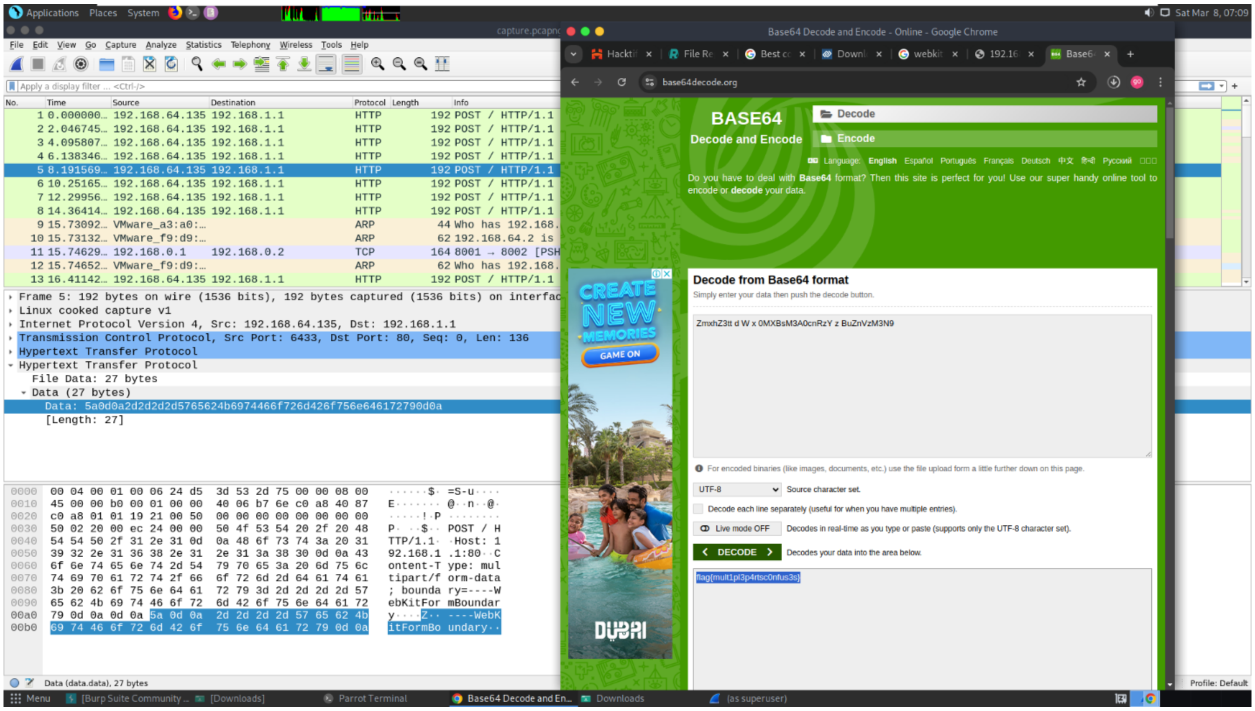
Description: The Network Forensics challenge involved analyzing captured network traffic to uncover hidden data. By carefully examining packet details, patterns, and communication flows, the investigation led to the successful extraction of the concealed flag.

Challenge Overview: Analyzed the provided file using Wireshark, focusing on HTTP POST method packets. Noticed a pattern where each packet contained a changing alphabet. Combined these characters and decoded the resulting string using a Base64 decoder to successfully extract the flag.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Examined the provided file in Wireshark to analyze network traffic.
2. **Input Validation Testing:** Reviewed the data packets, particularly focusing on HTTP POST requests for any unusual patterns.
3. **Directory Enumeration:** Identified recurring packets with varying data values, indicating potential encoded information.
4. **Exploitation:** Collected the changing alphabets observed in the HTTP POST packets, then combined them to form a meaningful string.
5. **Flag Retrieval:** Decoded the combined string using an online Base64 decoder to successfully retrieve the flag.

Flag: flag{mult1pl3p4rtsc0nfus3s}



```
No. Time        Source         Destination    Protocol Length Info
  1 0.000000…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  2 2.046745…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  3 4.095807…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  4 6.138346…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  5 8.191569…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  6 10.25165…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  7 12.29956…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  8 14.36414…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
  9 15.73092…   VMware_a3:a0:…                ARP         44 Who has 192.168.
 10 15.73132…   VMware_f9:d9:…                ARP         62 192.168.64.2 is
 11 15.74629…   192.168.0.1    192.168.0.2    TCP        164 8001 → 8002 [PSH
 12 15.74652…   VMware_f9:d9:…                ARP         62 Who has 192.168.
 13 16.41142…   192.168.64.135 192.168.1.1    HTTP       192 POST / HTTP/1.1
```

```
› Frame 5: 192 bytes on wire (1536 bits), 192 bytes captured (1536 bits) on interfac
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 192.168.64.135, Dst: 192.168.1.1
› Transmission Control Protocol, Src Port: 6433, Dst Port: 80, Seq: 0, Len: 136
› Hypertext Transfer Protocol
˅ Hypertext Transfer Protocol
    File Data: 27 bytes
  ˅ Data (27 bytes)
      Data: 5a0d0a2d2d2d2d5765624b6974466f726d426f756e646172790d0a
      [Length: 27]
```

```
0000  00 04 00 01 00 06 24 d5  3d 53 2d 75 00 00 08 00   ······$· =S-u····
0010  45 00 00 b0 00 01 00 00  40 06 b7 6e c0 a8 40 87   E······· @··n··@·
0020  c0 a8 01 01 19 21 00 50  00 00 00 00 00 00 00 00   ·····!·P ········
0030  50 02 20 00 ec 24 00 00  50 4f 53 54 20 2f 20 48   P· ··$·· POST / H
0040  54 54 50 2f 31 2e 31 0d  0a 48 6f 73 74 3a 20 31   TTP/1.1· ·Host: 1
0050  39 32 2e 31 36 38 2e 31  2e 31 3a 38 30 0d 0a 43   92.168.1 .1:80··C
0060  6f 6e 74 65 6e 74 2d 54  79 70 65 3a 20 6d 75 6c   ontent-T ype: mul
0070  74 69 70 61 72 74 2f 66  6f 72 6d 2d 64 61 74 61   tipart/f orm-data
0080  3b 20 62 6f 75 6e 64 61  72 79 3d 2d 2d 2d 2d 57   ; bounda ry=----W
0090  65 62 4b 69 74 46 6f 72  6d 42 6f 75 6e 64 61 72   ebKitFor mBoundar
00a0  79 0d 0a 0d 0a 5a 0d 0a  2d 2d 2d 2d 57 65 62 4b   y····Z·· ----WebK
00b0  69 74 46 6f 72 6d 42 6f  75 6e 64 61 72 79 0d 0a   itFormBo undary··
```

Data (data.data), 27 bytes

Do you have to deal with **Base64** format? Then this site is perfect for you! Use our super handy online tool to encode or **decode** your data.

Decode from Base64 format

Simply enter your data then push the decode button.

```
ZmxhZ3tt d W x 0MXBsM3A0cnRzY z BuZnVzM3N9
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

□ Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

```
flag{mult1pl3p4rtsc0nfus3s}
```

Category: Reverse Engg

Sub-Category: Lost in the Past

Description: The Reverse Engg challenge involved analyzing a compiled binary file to uncover hidden logic and extract the encoded flag by understanding the program's internal structure and behavior.

Challenge Overview: The Lost in the Past challenge required inspecting files within a provided zipped folder. A cipher text hidden in one of the files was decoded using "dencode.com," revealing the FLAG.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Examined the provided zipped folder and identified multiple files inside.
2. **Input Validation Testing:** Checked the contents of the files for suspicious patterns or encoded text.
3. **Directory Enumeration:** Explored the file structure within the zipped folder to ensure no hidden files were missed.
4. **Exploitation:** Identified a file containing cipher text and decoded it using "dencode.com."
5. **Flag Retrieval:** Successfully decoded the text to reveal the FLAG.

Flag: flag{t00_much_rev3rs1ng}

Category: Reverse Engg

Sub-Category: Decrypt Quest

Description: The Reverse Engg challenge involved analyzing a compiled binary file to uncover hidden logic and extract the encoded flag by understanding the program's internal structure and behavior.

Challenge Overview: The challenge involved extracting a ZIP file containing a text string that, when decoded using Base64, revealed a Java program. Running the program required an input derived from a hint in the provided Drive URL. The encoded string from the Drive URL was decoded using a Brainfuck decoder, ultimately revealing the FLAG.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Downloaded the provided ZIP file and examined its contents.
2. **Input Validation Testing:** Identified a text file within the ZIP folder containing a Base64 encoded string.
3. **Directory Enumeration:** Found a Drive URL mentioned as a hint, indicating additional information.
4. **Exploitation:** Decoded the Base64 string to reveal a Java program. Ran the Java program, which required an input. Decoded the encoded string from the Drive URL using a Brainfuck decoder to derive the required input.
5. **Flag Retrieval:** Successfully entered the decoded value into the Java program, revealing the FLAG.

Flag: flag{hjwilj111970djs}

Category: OSINT

Sub-Category: Time Machine

Description: The OSINT challenge involved gathering publicly available information through various online platforms, analyzing clues to uncover hidden data, ultimately leading to the FLAG.

Challenge Overview: The Time Machine challenge required investigating archived web data. By searching 'Mr. TrojanHunt travel time' on Google and exploring results on Archive.org, the FLAG was located inside a secret.txt file in the archive links.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Conducted a Google search using the keywords 'Mr. TrojanHunt travel time'.
2. **Input Validation Testing:** Analyzed search results to identify relevant links and references.
3. **Directory Enumeration:** Explored Archive.org for potential archived content related to the search query.
4. **Exploitation:** Accessed the links: https://archive.org/details/secret_202103.
5. **Flag Retrieval:** Found the secret.txt file in the directory path https://dn790008.ca.archive.org/0/items/secret_202103/secret.txt containing the FLAG.

Flag: flag{Tr0j3nHunt_t1m3_tr4v3l}

Category: OSINT

Sub-Category: Snapshot Whispers

Description: The OSINT challenge involved gathering publicly available information through various online platforms, analyzing clues to uncover hidden data, ultimately leading to the FLAG.

Challenge Overview: Revealed the FLAG by identifying the provided image using Google Lens, which pointed to the Sydney Opera House. Further investigation on Google Maps, filtering reviews with the keyword 'concert hall', provided the necessary clues.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Analyzed the provided image and used Google Lens for identification.

2. **Input Validation Testing:** Verified the identified location as the Sydney Opera House.

3. **Directory Enumeration:** Conducted a Google Maps search for the Sydney Opera House and filtered reviews using the keyword 'concert hall'.

4. **Exploitation:** Explored filtered reviews to gather relevant clues and information.

5. **Flag Retrieval:** Identified the required details in one of the reviews, successfully obtaining the FLAG.

Flag: flag{jeffrey_seidman}

Category: Crypto

Sub-Category: Wh@t7he####

Description: A cryptography-based challenge requiring decoding an encrypted text using specific ciphers and decryption techniques to uncover the hidden FLAG.

Challenge Overview: The challenge involved decoding an encoded string using the "ReverseFuck decoder," which successfully revealed the hidden FLAG.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Analyzed the provided encoded string to identify the type of cipher used.
2. **Input Validation Testing:** Tested various decoding methods to determine the correct decryption technique.
3. **Directory Enumeration:** Explored available online decoding tools for potential solutions.
4. **Exploitation:** Used the "ReverseFuck decoder" to decode the given string.
5. **Flag Retrieval:** Successfully obtained the FLAG from the decoded output.

Flag: flag{R3vers3ddd_70_g3t_m3}

Category: Crypto

Sub-Category: Success Recipe

Description: A cryptography-based challenge requiring decoding an encrypted text using specific ciphers and decryption techniques to uncover the hidden FLAG.

Challenge Overview: Solved the challenge by fixing syntax issues in the provided Chef code, compiling it using an online Chef compiler, and decoding the resulting output with a BrainFuck decoder to retrieve the FLAG.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** Reviewed the provided Chef code and identified multiple syntax errors.
2. **Input Validation Testing:** Analyzed the structure of the Chef code to ensure correct formatting and logic.
3. **Directory Enumeration:** No directory enumeration was required for this challenge.
4. **Exploitation:** Corrected the grammatical and syntax errors in the Chef code and compiled it using an online Chef compiler.
5. **Flag Retrieval:** Decoded the compiled output using an online BrainFuck decoder, successfully obtaining the FLAG.

Flag: flag{y0u_40+_s3rv3d!}