CTF Report

Full Name: Aurelia Anthony
Program: HCS - Penetration Testing 1-Month Internship
Date: 07/03/2025

Category: Web 2.0

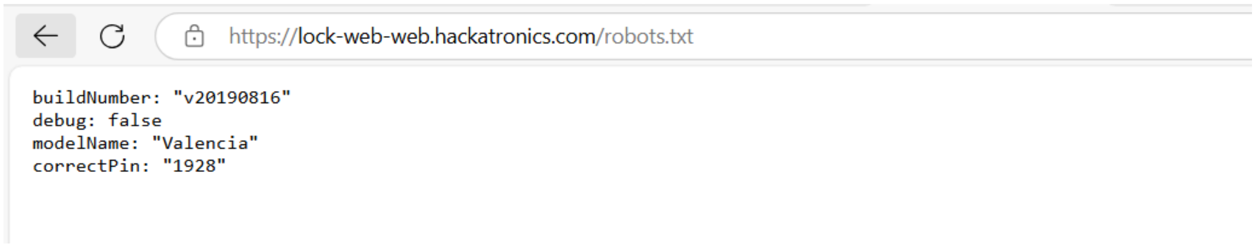
Sub - Category: Lock Web

Description: Lock Web: It is critical to use proper content discovery methodology when testing websites. This is not usually similar to dirbuster or other bruteforcing techniques.

Challenge Overview: This challenge requires o to use advanced content discovery techniques to reveal buried content on a web application. Unlike standard brute-force methods, one will need to take a more deliberate approach to identifying and exploiting hidden endpoints, which will reveal the flag.

Steps for Finding the Flag:

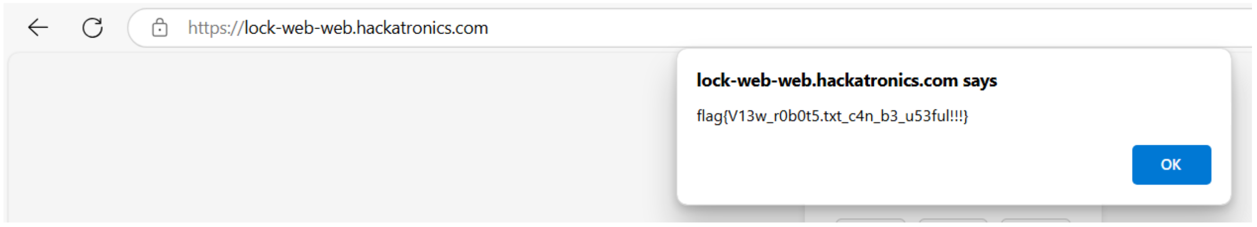
1. **Initial Reconnaissance:** A link to a webpage was provided. Begin by studying the web application's functionality and structure. It includes a pin code input page. Look for input fields, forms, URLs, and other potentially exploitable features.
2. **Input Validation Testing:** Attempt to inject malicious payloads into the input fields and URL parameters.
3. **Directory Enumeration:** Explore the web application's directories and endpoints to find hidden pages or functions that could lead to the flag. Use */robots.txt* as the URL parameter which serves as a dictionary brute forcing. This file tells search engine crawlers which pages of a website they can access.



https://lock-web-web.hackatronics.com/robots.txt

```
buildNumber: "v20190816"
debug: false
modelName: "Valencia"
correctPin: "1928"
```

4. **Exploitation:** Enter the Pin into the input field
5. **Flag Retrieval:** The flag is then discovered. Capture it and document it for submission.



https://lock-web-web.hackatronics.com

lock-web-web.hackatronics.com says

flag{V13w_r0b0t5.txt_c4n_b3_u53ful!!!}

Flag: *flag{V13w_r0b0t5.txt_c4n_b3_u53ful!!!}*

Sub - Category: The World

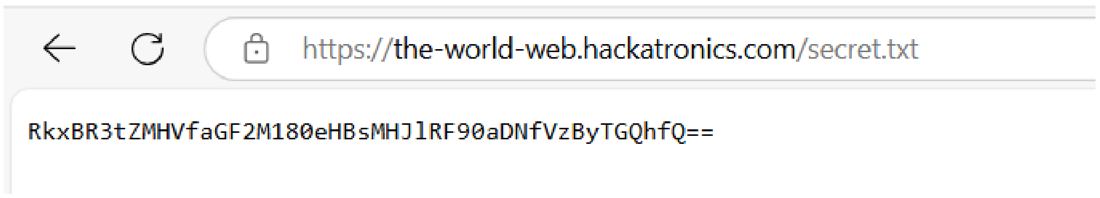
Description: Welcome to "The World" Challenge! You've arrived at a webpage that says, "Hello World!" Looks simple, doesn't it? However, there is more to it than meets the eye. Your aim is to delve deep into this website to discover hidden passageways and the flag.

Challenge Overview: This challenge requires you to examine a seemingly simple webpage that states, "Hello World!" Your mission is to discover hidden routes and surprises across the site. Discovering and decrypting the /secret.txt file will reveal the flag and complete the task.

Steps for Finding the Flag:

1. **Initial Reconnaissance:** A link to a webpage was provided. Begin by studying the web application's functionality and structure. It includes a pin code input page. Look for input fields, forms, URLs, and other potentially exploitable features.

2. **Input Validation Testing:** Attempt to inject malicious payloads into the URL parameters.
3. **Directory Enumeration:** Explore the web application's directories and endpoints to find hidden pages or functions that could lead to the flag. Use *secret.txt* as the URL parameter. This file tells search engine crawlers which pages of a website they can access.
4. **Exploitation:** Once the */secret.txt* file is entered into the URL, it returns an encrypted code. Decrypt it, and the flag will be shown.



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

```
RkxBR3tZMHVfaGF2M180eHBsMHJlRF90aDNfVzByTGQhfQ==
```

5. **Flag Retrieval:** Once the flag is discovered, capture and document it for submission.

Flag: *FLAG{Y0u_hav3_4xpl0reD_th3_W0rLd!}*

Category: Network Forensics

Sub- Category: Corrupted

Description: A shared file has been corrupted and requires decoding to restore it to a viewable format.

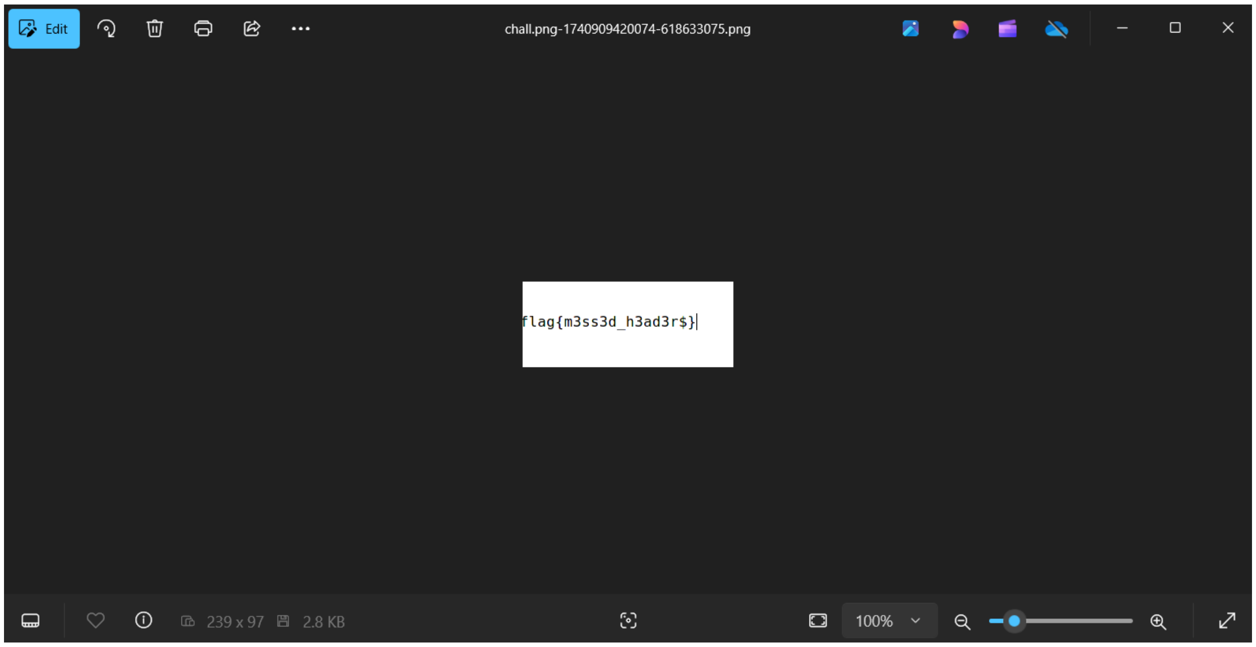
Challenge Overview: In this challenge, you are presented with a corrupted file that needs to be decoded into a viewable format. Using tools like hexedit, you must analyze and repair the file to uncover the hidden flag.

Steps for Finding the Flag:

1. **File Analysis:** Use an online hex editor like hexedit to study the corrupted file's raw hexadecimal content. Search for anomalies or missing file headers.

2. **Header Correction:** Find the erroneous or missing file header. Replace the first line of binary code with the standard PNG file header, which is: 89 50 4E 47 0D 0A 1A 0A.

3. **Save and Verify:** After editing the file, save it and try opening it as a PNG image to ensure its integrity.
4. **Flag Discovery:** After successfully fixing the file, open it to see the hidden flag.



```
flag{m3ss3d_h3ad3r$}
```

Flag: *flag{m3ss3d_h3ad3r$}*

Sub - Category: Shadow Web

Description: Uncover hidden data within the complex network of protocols. This MULTIverse of packets has some Form Data that may expose the secrets of the Web. Try to uncover the scattered secrets in order to obtain a flag.

Challenge Overview: In this challenge, you are tasked with uncovering hidden data within a complex network of protocols. By analyzing HTTP packets in a provided PCAP file, you'll piece together scattered secrets to decode the flag.

Steps for Finding the Flag:

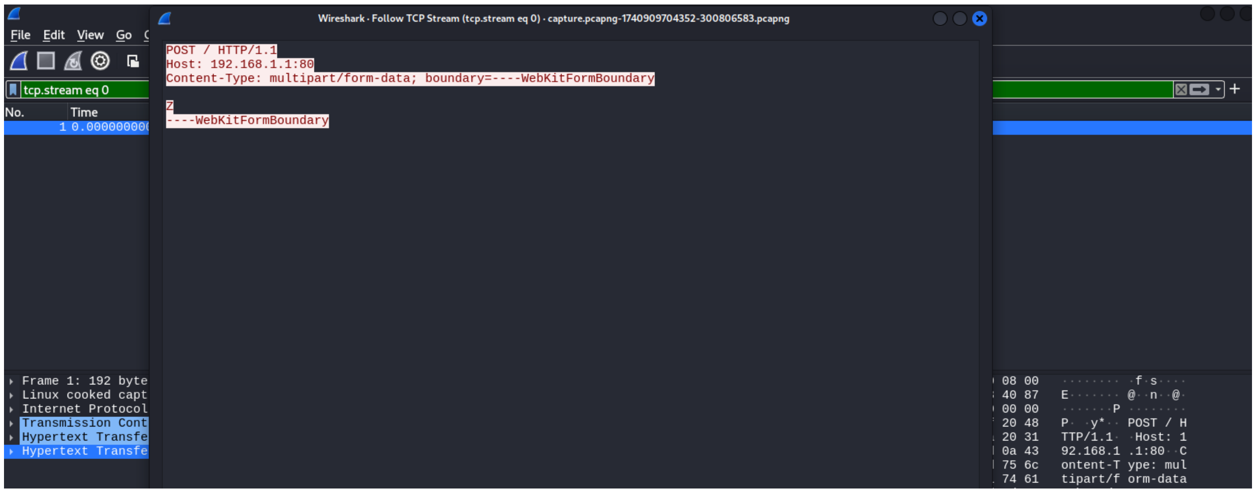
1. **Packet Analysis:** To analyze the collected traffic, open the given PCAP file in Wireshark and apply the HTTP packet filter.

2. **Data Extraction:** Examine each HTTP packet to detect and extract hidden characters or encoded data buried in the payload.

3. **Cipher Compilation:** Convert the retrieved letters into a cohesive cipher or encoded string.

4. **Decoding:** Use a Base64 decoder to decipher the constructed encryption and discover the hidden flag.

5. **Flag Discovery:** Obtain and document the decoded flag for submission.


Wireshark · Follow TCP Stream (tcp.stream eq 0) · capture.pcapng-1740909704352-300806583.pcapng
File Edit View Go
tcp.stream eq 0
No. Time

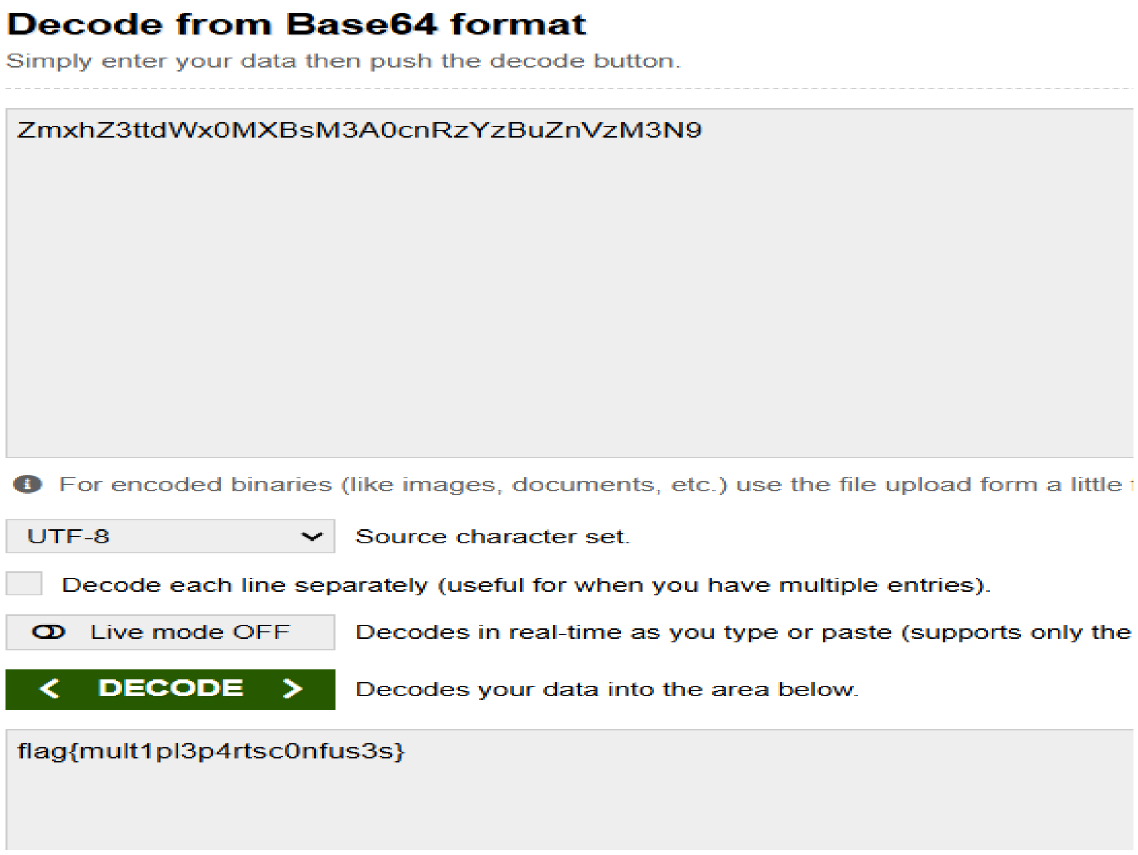
```
POST / HTTP/1.1
Host: 192.168.1.1:80
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary

Z
----WebKitFormBoundary
```

```
1 0.000000000
```

```
Frame 1: 192 byte
Linux cooked capt
Internet Protocol
Transmission Cont
Hypertext Transfe
Hypertext Transfe
```

```
08 00   ········ ·f·s····
40 87   E······· @··n··@·
00 00   ·······P ········
20 48   P·  ·y*·· POST / H
20 31   TTP/1.1·  ·Host: 1
0a 43   92.168.1 .1:80··C
75 6c   ontent-T ype: mul
74 61   tipart/f orm-data
```



Decode from Base64 format

Simply enter your data then push the decode button.

```
ZmxhZ3ttdWx0MXBsM3A0cnRzYzBuZnVzM3N9
```

For encoded binaries (like images, documents, etc.) use the file upload form a little

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

DECODE — Decodes your data into the area below.

```
flag{mult1pl3p4rtsc0nfus3s}
```

Flag: *flag{mult1pl3p4rtsc0nfus3s}*

Category: Reverse Engg

Sub - Category: Lost in the Past

Description: In this challenge, you are required to reverse engineer a set of files to find a hidden flag. To discover prior hidden flag, you must extract and analyze the compressed file and decode the underlying cipher.

Challenge Overview: This challenge tasks you to extract and evaluate an XML document from a zip file. The XML contains a cipher that must be decoded using the right encryption type. After decoding, the flag will be revealed.

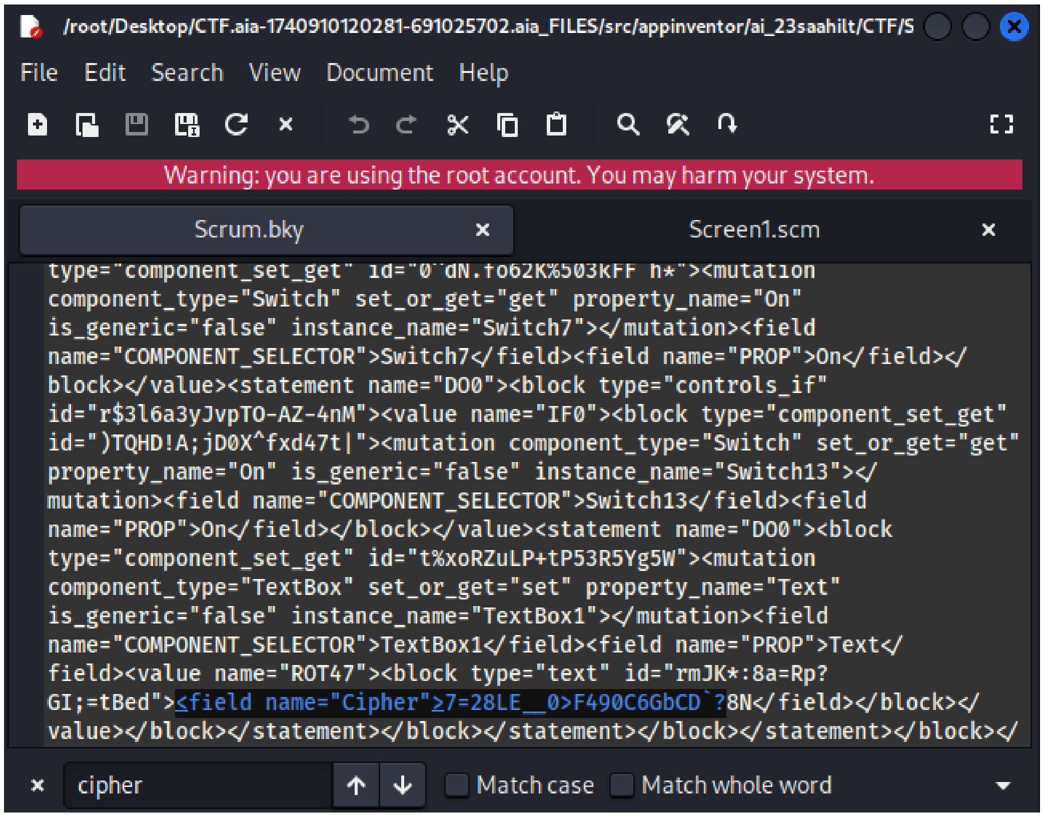
Steps for Finding the Flag:

1. **File Extraction:** Unzip the provided zipped file and examine the files in the folder.

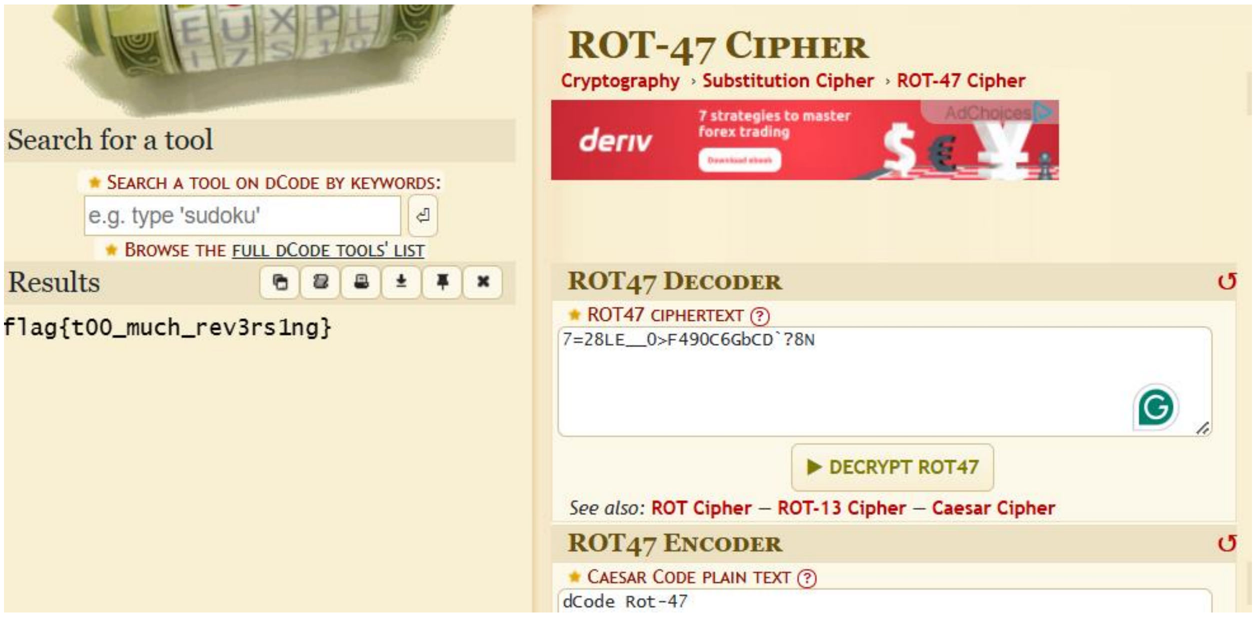
2. **Cipher Identification:** Find the XML file and identify the cipher included in it. Determine the encryption type using tools such as dcode.fr (for example, ROT-47).

3. **Cipher Decoding:** To disclose the concealed flag, decode the specified cipher using the proper method (for example, ROT-47 decoder).

4. **Flag Retrieval:** Capture and document the decoded flag for submission.


/root/Desktop/CTF.aia-1740910120281-691025702.aia_FILES/src/appinventor/ai_23saahilt/CTF/S
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
Scrum.bky
Screen1.scm
cipher
Match case
Match whole word

```
type="component_set_get" id="0`dN.fo62K%503kFF h*"><mutation
component_type="Switch" set_or_get="get" property_name="On"
is_generic="false" instance_name="Switch7"></mutation><field
name="COMPONENT_SELECTOR">Switch7</field><field name="PROP">On</field></
block></value><statement name="DO0"><block type="controls_if"
id="r$3l6a3yJvpTO-AZ-4nM"><value name="IF0"><block type="component_set_get"
id=")TQHD!A;jD0X^fxd47t|"><mutation component_type="Switch" set_or_get="get"
property_name="On" is_generic="false" instance_name="Switch13"></
mutation><field name="COMPONENT_SELECTOR">Switch13</field><field
name="PROP">On</field></block></value><statement name="DO0"><block
type="component_set_get" id="t%xoRZuLP+tP53R5Yg5W"><mutation
component_type="TextBox" set_or_get="set" property_name="Text"
is_generic="false" instance_name="TextBox1"></mutation><field
name="COMPONENT_SELECTOR">TextBox1</field><field name="PROP">Text</
field><value name="ROT47"><block type="text" id="rmJK*:8a=Rp?
GI;=tBed"><field name="Cipher">7=28LE__0>F490C6GbCD`?8N</field></block></
value></block></statement></block></statement></block></
```


Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type 'sudoku'

BROWSE THE FULL DCODE TOOLS' LIST

Results

```
flag{t00_much_rev3rs1ng}
```

ROT-47 CIPHER

Cryptography › Substitution Cipher › ROT-47 Cipher

ROT47 DECODER

ROT47 CIPHERTEXT

```
7=28LE__0>F490C6GbCD`?8N
```

DECRYPT ROT47

See also: ROT Cipher – ROT-13 Cipher – Caesar Cipher

ROT47 ENCODER

CAESAR CODE PLAIN TEXT

```
dCode Rot-47
```

Flag: *flag{t00_much_rev3rsing}*

Category: OSINT

Sub - Category: Time Machine

Description: Mr. TrojanHunt has the ability to travel in time. He is hiding a highly classified file from the government. Can you assist NIA discover TrojanHunt's secrets?

Challenge Overview: In this task, you will use OSINT techniques to uncover information on Mr. TrojanHunt and his time-traveling activities. By analyzing search results and downloading a certain file, you can discover the hidden flag.

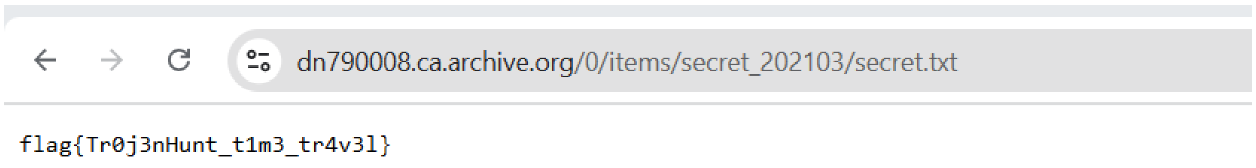
Steps for Finding the Flag:

1. **Initial Search:** Use a search engine (such as Google) to hunt up information about "Mr. TrojanHunt Time Machine."

2. **Archive Discovery:** Locate and access the appropriate webpage or archive that has information about Mr. TrojanHunt.

3. **File download:** Download the provided text file from the website.

4. **Flag Retrieval:** Open the text file to locate and record the hidden flag.



```
dn790008.ca.archive.org/0/items/secret_202103/secret.txt

flag{Tr0j3nHunt_t1m3_tr4v3l}
```

Flag: *flag{Tr0j3nHunt_t1m3_tr4v3l}*

Sub - Category: Snapshot Whispers

Description: In this challenge, an image serves as a hint to uncover a hidden flag. By identifying the location in the image and analyzing related information, you'll piece together the clues to reveal the flag.

Challenge Overview: An image of the Sydney Opera House is used as a hint. The job is to identify the location, analyze reviews or related information, and extract the flag, which is a specific reviewer's first and last name in the format flag{Firstname_Lastname}.

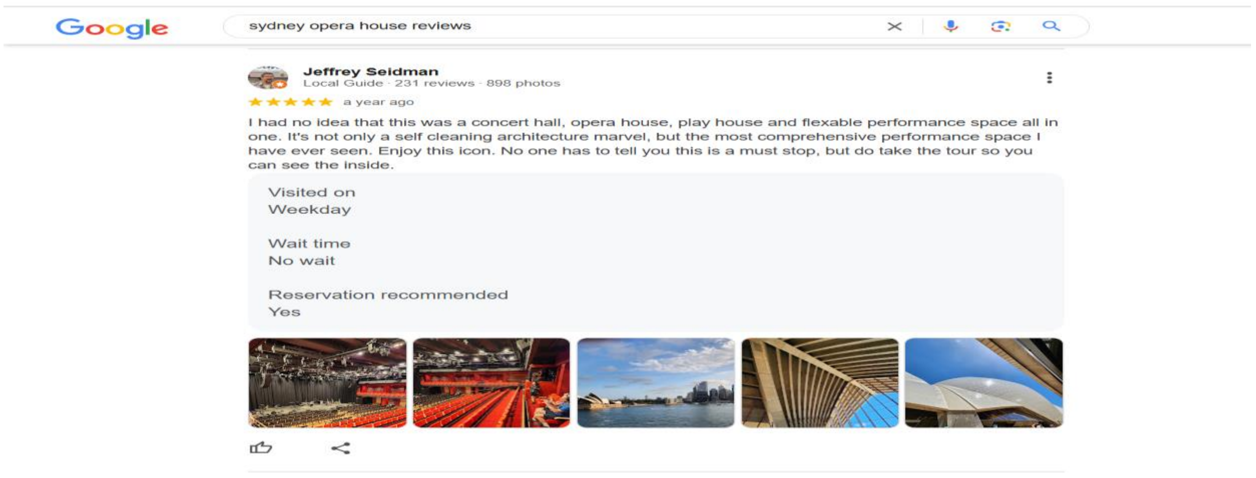
Steps for Finding the Flag:

1. **Image Analysis:** Identify the location featured in the provided image as the Sydney Opera House.

2. **Review Investigation:** Search for reviews of the Sydney Opera House and locate the image in a year-old review.

3. **Flag Extraction:** Extract the first and last name of the reviewer mentioned in the review description.

4. **Flag Formatting:** Format the reviewer's name as flag{Firstname_Lastname} to discover the flag.


sydney opera house reviews

Jeffrey Seidman
Local Guide · 231 reviews · 898 photos
★★★★★ a year ago

I had no idea that this was a concert hall, opera house, play house and flexable performance space all in one. It's not only a self cleaning architecture marvel, but the most comprehensive performance space I have ever seen. Enjoy this icon. No one has to tell you this is a must stop, but do take the tour so you can see the inside.

Visited on
Weekday

Wait time
No wait

Reservation recommended
Yes

Flag: flag{Jeffrey_Seidman}

Category: Crypto

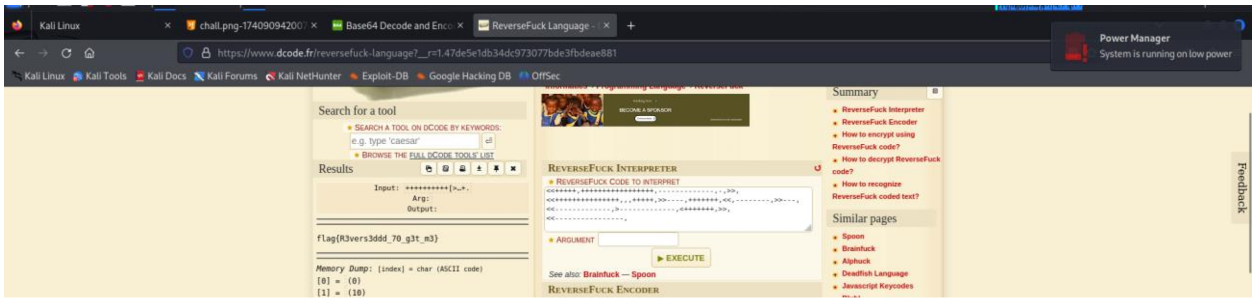
Sub - Category: Wh@t7he####

Description: In this challenge, one is given a file containing encoded data. Your objective is to decode the input with the necessary tools and procedures in order to discover the hidden flag.

Challenge Overview: You can download a file containing encoded data. Using a reversefuck decoder, you can decode the input and discover the hidden flag.

Steps for Finding the Flag:

1. **File download:** Download the given file, which contains the encoded input.
2. **Input extraction:** Copy the encoded input from the file.
3. **Cipher Identification:** Use dcode.fr to determine the type of cipher (for example, Reversefuck).
4. **Decoding:** Use a reversefuck decoder to decode the incoming data.
5. **Flag Discovery:** Following decoding, the concealed flag will be revealed.


https://www.dcode.fr/reversefuck-language?__r=1.47de5e1db34dc973077bde3fbdeae881

ReverseFuck Interpreter

```
<<++++++,++++++++++++++++++,.............,.,>>,
<<++++++++++++++++,.,+++++,>>----,+++++++,<<,--------,>>---,
<<.............,>............,<+++++++,>>,
<<...............,
```

Results

```
Input: ++++++++++[>_+.
  Arg:
Output:

flag{R3vers3ddd_70_g3t_m3}

Memory Dump: [index] = char (ASCII code)
[0] =  (0)
[1] =  (10)
```

Flag: *flag{R3vers3ddd_70_g3t_m3}*

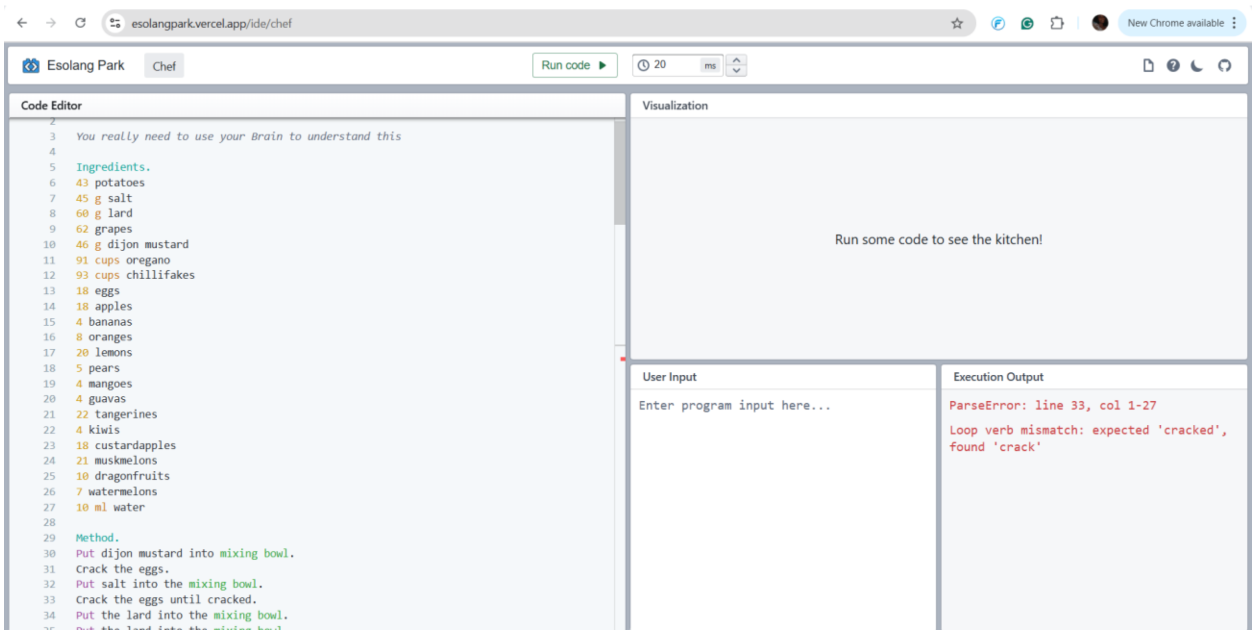
Sub - Category: Success Recipe

Description: In this challenge, you are provided with a file containing encoded data. Your task is to decode the input using an esolang decoder, identify the cipher type, and ultimately uncover the hidden flag.

Challenge Overview: A file is provided for download, containing encoded data. After rectifying any errors in the input, you must identify the encryption type and decode the data to disclose the flag.

Steps for Finding the Flag:

1. **File Download:** Download the given file, which contains the encoded input.
2. **Input Extraction:** Copy the encoded input from the file.
3. **Error Correction**: Input the data into an esolang decoder and rectify any errors that occur.
4. **Cipher Identification:** Use dcode.fr to determine the type of cipher (for example, Brainfuck).
5. **Decoding Process:** Use the proper decoder to decode the input (for example, the Brainfuck decoder).
6. **Flag Discovery:** Upon decoding, the concealed flag will be revealed.



esolangpark.vercel.app/ide/chef

Esolang Park | Chef | Run code | 20 ms

Code Editor

```
You really need to use your Brain to understand this

Ingredients.
43 potatoes
45 g salt
60 g lard
62 grapes
46 g dijon mustard
91 cups oregano
93 cups chillifakes
18 eggs
18 apples
4 bananas
8 oranges
20 lemons
5 pears
4 mangoes
4 guavas
22 tangerines
4 kiwis
18 custardapples
21 muskmelons
10 dragonfruits
7 watermelons
10 ml water

Method.
Put dijon mustard into mixing bowl.
Crack the eggs.
Put salt into the mixing bowl.
Crack the eggs until cracked.
Put the lard into the mixing bowl.
```

Visualization

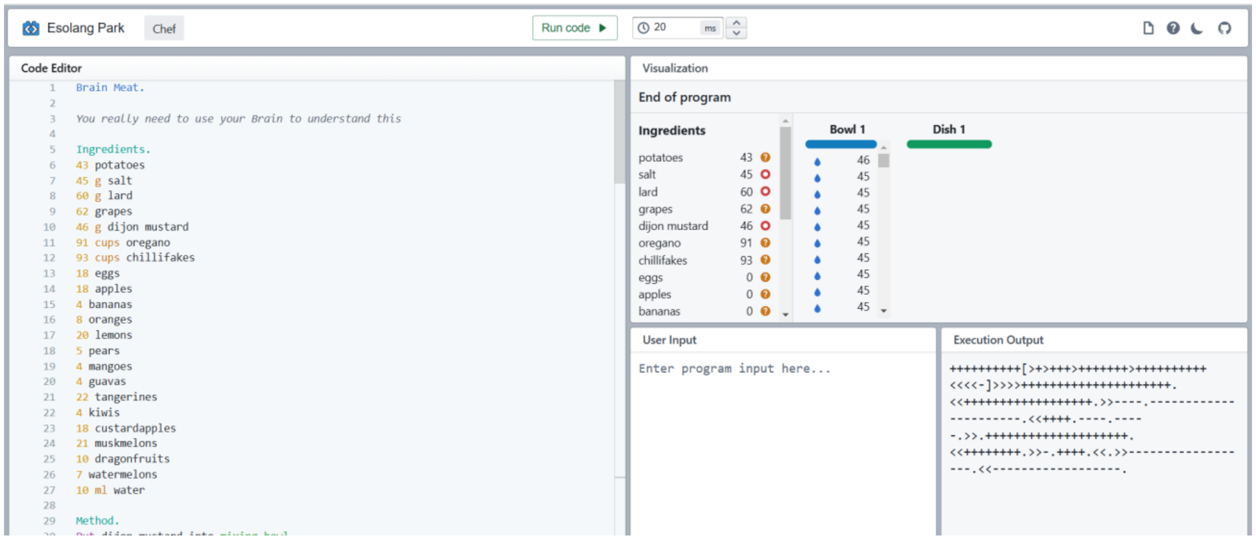
Run some code to see the kitchen!

User Input

Enter program input here...

Execution Output

```
ParseError: line 33, col 1-27
Loop verb mismatch: expected 'cracked', found 'crack'
```



Esolang Park | Chef | Run code | 20 ms

Code Editor

```
Brain Meat.

You really need to use your Brain to understand this

Ingredients.
43 potatoes
45 g salt
60 g lard
62 grapes
46 g dijon mustard
91 cups oregano
93 cups chillifakes
18 eggs
18 apples
4 bananas
8 oranges
20 lemons
5 pears
4 mangoes
4 guavas
22 tangerines
4 kiwis
18 custardapples
21 muskmelons
10 dragonfruits
7 watermelons
10 ml water

Method.
```

Visualization

End of program

Ingredients		Bowl 1
potatoes	43	46
salt	45	45
lard	60	45
grapes	62	45
dijon mustard	46	45
oregano	91	45
chillifakes	93	45
eggs	0	45
apples	0	45
bananas	0	45

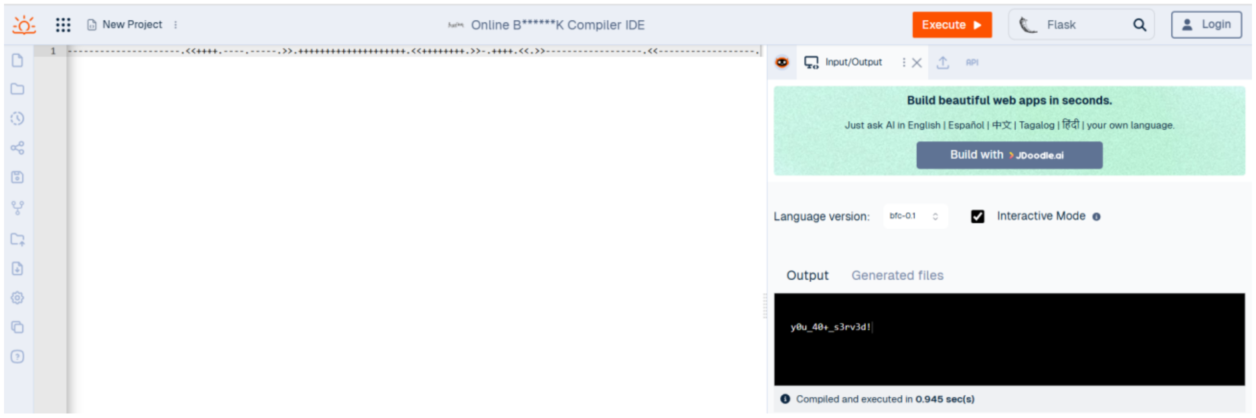
Dish 1

User Input

Enter program input here...

Execution Output

```
++++++++++[>+>+++>+++++++>++++++++++
<<<<-]>>>>+++++++++++++++++++++.
<<++++++++++++++++++.>>----.------------
----------.<<++++.----.----
-.>>.++++++++++++++++++++.
<<++++++++.>>-.++++.<<.>>---------------
---.<<------------------.
```



```
--------------------.<<++++.----.-----.>>.+++++++++++++++++++++.<<++++++++.>>-.++++.<<.>>------------------.<<------------------.
```

```
y0u_40+_s3rv3d!
```

Compiled and executed in **0.945 sec(s)**

Flag: *flag{ y0u_40+_s3rv3d!}*