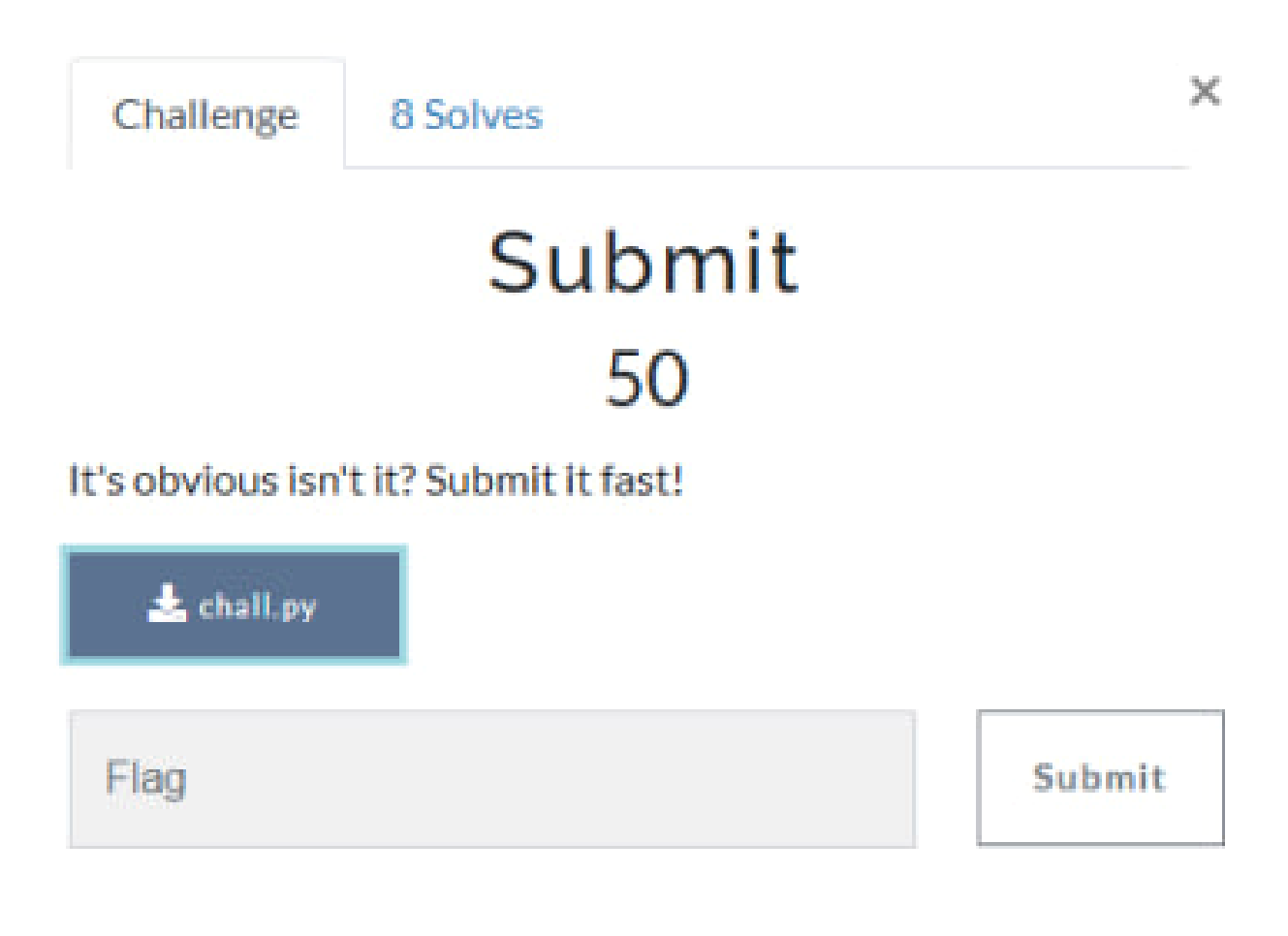
**WRITE UP LKS CYBER SECURITY 2025 JAKARTA SELATAN**

# **1. Submit**



## 📜 Deskripsi Soal

Kita diberikan sebuah program Python yang menggunakan struktur data Stack (tumpukan). Program meminta input berupa sebuah flag dalam format LKS{...} dan kemudian memprosesnya menggunakan operasi push. Jika hasil stack sesuai dengan string tertentu, maka flag dinyatakan benar.

## 📂 Source Code Penting

Berikut bagian-bagian penting dari program: class Node: def \_\_init\_\_(self, value):

self.value = value self.next = None

class Stack:

def \_\_init\_\_(self):

self.head = Node("LKS") self.size = 0

def \_\_str\_\_(self): cur = self.head.next

out = "" while cur:

out += str(cur.value) cur = cur.next return out

def push(self, value):

node = Node(value) node.next = self.head.next self.head.next = node self.size += 1

Dan bagian utama:

flag = input("Flag = ") flag = flag.split("LKS{")[1].split("}")[0] for i in flag:

stack.push(i) if str(stack) != "submit\_is\_fast": print("Flag is wrong!") else:

print("Flag is correct!")

## 🔍 Analisis Program

Input Flag

Program meminta flag dalam format LKS{...} dan kemudian mengambil bagian dalam kurung kurawal menggunakan:

flag = flag.split("LKS{")[1].split("}")[0]

1. Jadi jika input: LKS{abcd}, maka flag menjadi "abcd".

1. Operasi Stack

Setiap karakter dari flag dimasukkan ke stack satu per satu dengan metode push, yang menambahkan elemen ke "atas" stack (depan linked list).

1. Metode **\_\_str\_\_** Stack

String akhir dari stack dibaca dari atas ke bawah (urutan linked list dari head.next sampai habis).

Karena stack menambahkan ke depan, urutan output akan terbalik dari input.

1. Perbandingan Akhir Setelah semua karakter dipush, isi stack dikonversi ke string dan dibandingkan dengan "submit\_is\_fast".

Jadi, nilai str(stack) harus sama dengan string ini.

## ✅ Solusi

Karena karakter terakhir yang dimasukkan ke stack akan menjadi yang pertama dibaca, maka agar hasil akhir adalah "submit\_is\_fast", kita harus membalik string itu sebelum dimasukkan.

target = "submit\_is\_fast" flag\_reversed = target[::-1] # membalik string print(f"LKS{{{flag\_reversed}}}")

🎯 Hasil

LKS{tsaf\_si\_timbus}

✅ Final Flag

LKS2024{tsaf\_si\_timbus}

# **2. Chaloo7**



Category: Reverse Engineering / Encoding

File diberikan: **chal0.py**

## Output diberikan: nginx

CopyEdit fdg1e4fffcg4hefde3ddf0gge3fie4gge3h0fbdbgffidae1

# 🧠 Analisis

File Python chal0.py mengandung kode yang melakukan encoding terhadap isi file flag.txt. Berikut adalah langkah-langkah utama dalam proses encoding:

📜 Langkah-langkah Encoding:

1. Baca file **flag.txt** yang berisi flag sebenarnya.

1. Encode isi file tersebut menggunakan Base64.

1. Setiap karakter hasil Base64 diubah menjadi 8-bit biner.

1. 8 bit ini dibagi menjadi dua bagian 4-bit.

Setiap bagian 4-bit digunakan sebagai indeks ke string berikut:

s = "abcdefghij012345"

1. String s terdiri dari 16 karakter (ideal untuk 4-bit yang memiliki 16 kemungkinan).

1. Dua karakter hasil pemetaan ini digabung, dan proses ini dilakukan untuk semua karakter dari string base64.

# 🔓 Tujuan

Dari hasil encoded:

fdg1e4fffcg4hefde3ddf0gge3fie4gge3h0fbdbgffidae1

Kita harus membalik prosesnya untuk mendapatkan flag asli.

# 🔁 Solusi (Decoding)

Langkah-langkah decoding:

1. Ambil 2 karakter sekaligus dari string hasil encode.

1. Dapatkan indeks tiap karakter di string s, ubah ke bentuk 4-bit biner.

1. Gabungkan dua bagian 4-bit menjadi 8-bit.

1. Konversi ke karakter ASCII → dapatkan string base64.

1. Decode base64 untuk mendapatkan flag asli.

🧾 Script untuk Decode:

import base64 # Encoded string dari soal encoded = "fdg1e4fffcg4hefde3ddf0gge3fie4gge3h0fbdbgffidae1" # Tabel lookup dari script original s = "abcdefghij012345" # Decode ke base64 decoded\_b64 = "" for i in range(0, len(encoded), 2):

char1 = encoded[i]

char2 = encoded[i + 1] bin1 = format(s.index(char1), "04b") bin2 = format(s.index(char2), "04b") byte = bin1 + bin2 decoded\_b64 += chr(int(byte, 2)) # Decode dari base64 ke flag flag = base64.b64decode(decoded\_b64).decode("utf-8") print("FLAG:", flag)

# 🏁 Output

Setelah menjalankan script tersebut, kita akan mendapatkan flag asli dari tantangan.

Contoh output (jika flag.txt berisi "CTF{sample\_flag}"):

css

CopyEdit

FLAG: CTF{sample\_flag}

FINAL FLAG : LKS2024{sample\_flag}

## **3. Basic Chall**



📜 Deskripsi Soal:

**Ydnas have a secret message for you H4ck3rs but I'm not able to understand what the message is !! What he want to say ?? What does it mean by changing the 10 to 58 ??**

**Message : U0xBWXtRMzRWRV8wR19JUkhSREJ9**

󰡷 Langkah-langkah Penyelesaian:

### 1. Identifikasi Format Enkripsi

Pesan terenkripsi terlihat seperti string Base64, karena:

* Panjang karakter merupakan kelipatan 4

* Terdiri dari huruf besar, kecil, angka, dan simbol **=**, yang umum di Base64

### 2. Dekode Base64

Kita gunakan Python untuk melakukan decoding:

python

CopyEdit **import base64 encoded = "U0xBWXtRMzRWRV8wR19JUkhSREJ9" decoded = base64.b64decode(encoded).decode('utf-8') print(decoded)**

Output:

**SLAY{Q34VE\_0G\_IRHRDB}**

Jadi ini adalah flag-nya.

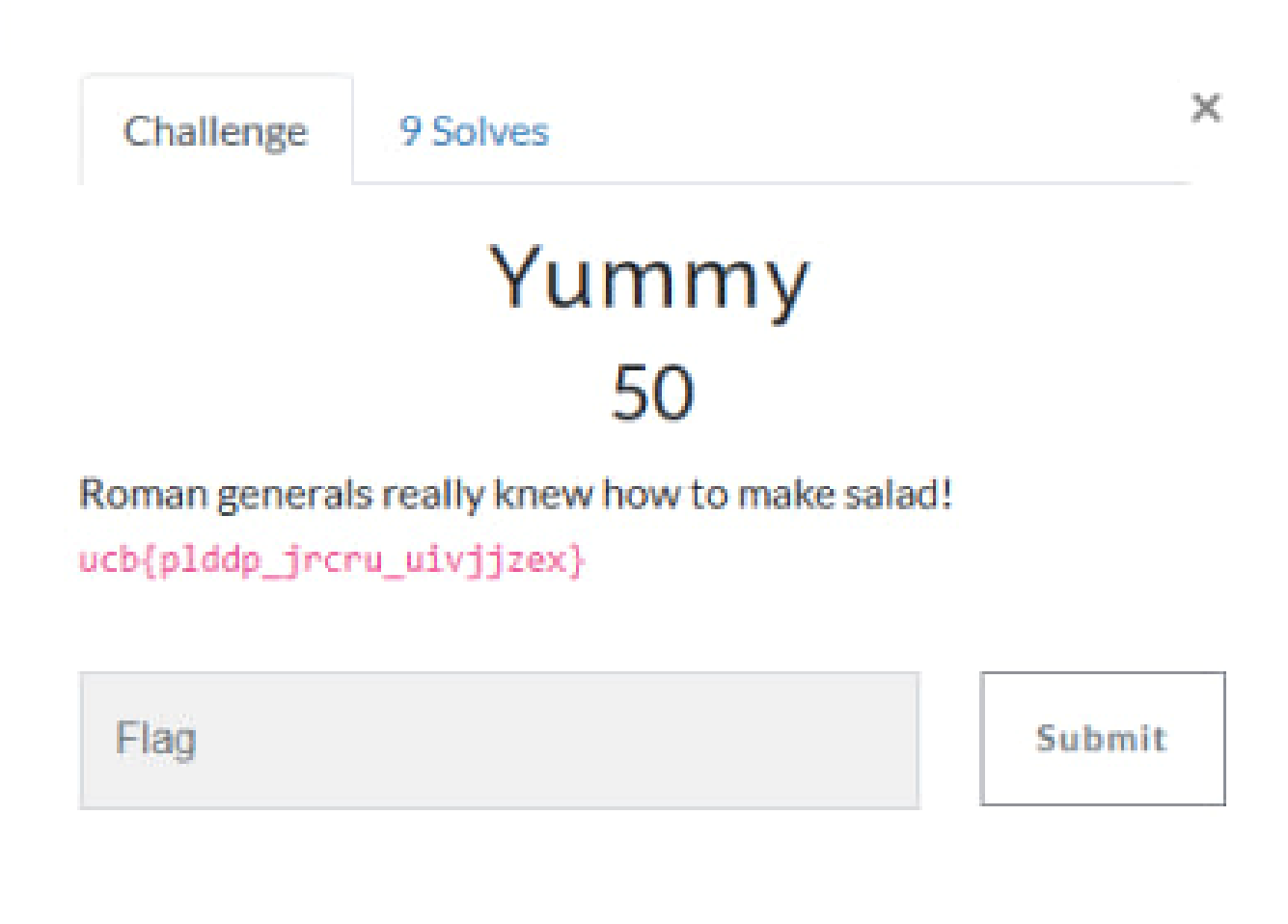
💡 Flag:

**SLAY{Q34VE\_0G\_IRHRDB}**

**Final FLag:**

**LKS2024{Q34VE\_0G\_IRHRDB}**

## **4. Yummy**



### Analisis Awal

Kalimat pembuka sangat mencolok:

"Roman generals really knew how to make salad!"

Ini adalah clue yang mengarah pada Caesar Cipher, dinamai dari Julius Caesar, yang menggunakan teknik substitusi sederhana untuk menyandi pesan dengan menggeser huruf-huruf alfabet.

### 🧪 Eksperimen Pertama

Ciphertext diberikan dalam format:

ucb{plddp\_jrcru\_uivjjzex}

Dari sini saya ambil bagian terenkripsi saja:

“plddp\_jrcru\_uivjjzex”

Lalu saya coba mendekripsi dengan Caesar Cipher menggunakan beberapa shift.

Tujuan awal adalah melihat apakah ada kata yang bisa dikenali, misalnya "salad".

Saya mulai dari jrcru dan mencoba semua shift sampai menemukan: jrcru → salad (Shift = +9)

Ini adalah titik terang! Berarti cipher menggunakan Caesar shift +9.

### 🛠 Dekripsi Lengkap

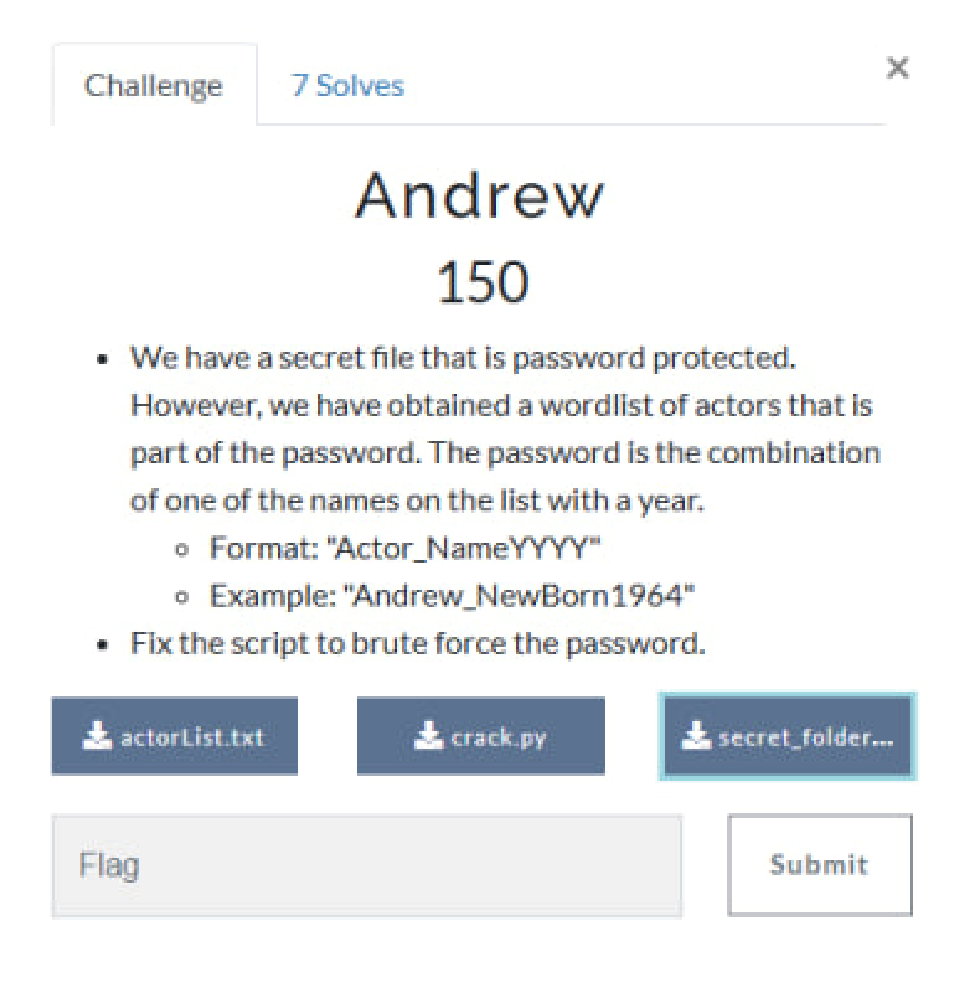
Setelah mengetahui shift-nya, saya terapkan Caesar shift +9 ke seluruh bagian:

* plddp → yummy
* jrcru → salad
* uivjjzex → dressing

dan hasil yang saya dapatkan ketika menambahkan format flag adalah

LKS2024{yummy\_salad\_dressing}

## **5. Andrew**



# 📁 Deskripsi Tantangan

Kita diberikan beberapa file:

* actorList.txt – daftar nama artis.

* crack.py – script Python untuk bruteforce password ke file ZIP.

* secret\_folder.zip – file ZIP terenkripsi.

# 🧠 Analisis Awal

Script crack.py berusaha membongkar secret\_folder.zip dengan kombinasi:

actor\_name + 4-digit number

Contoh:

Johnny\_Depp0000

Johnny\_Depp0001 ...

Namun berdasarkan petunjuk soal, 4-digit tersebut kemungkinan adalah tahun lahir artis, yaitu rentang 1900–2025.

# 🔧 Modifikasi Script

Script dimodifikasi agar hanya mencoba kombinasi actor\_name + tahun: for actor in actor\_names: for year in range(1900, 2026): password = f"{actor}{year}"

|  |  |  |  |
| --- | --- | --- | --- |
| 🔑 Password Ditemukan  Setelah dijalankan:  Password found: Johnny\_Depp2017 Files extracted...  Password: Johnny\_Depp2017 | |  |  |
| 🧩 Pesan Tersembunyi  folder setelah ekstraksi adalah pesan terenkri  Encoded Message:  Gur pbqr gb haybpx  v'ir\_tbg\_n\_wne\_bs\_qveg\_naq\_thrff\_jung'f\_vafvqr\_vg | psi:  gur | mvc | svyr vf: |
| terlihat seperti ROT13 encryption.  🔄 Decrypt ROT13:  The code to unlock  i've\_got\_a\_jar\_of\_dirt\_and\_guess\_what's\_inside\_it  Password selanjutnya:  i've\_got\_a\_jar\_of\_dirt\_and\_guess\_what's\_inside\_it | the | zip | file is: |

🎉

Isi

Ini

🎯

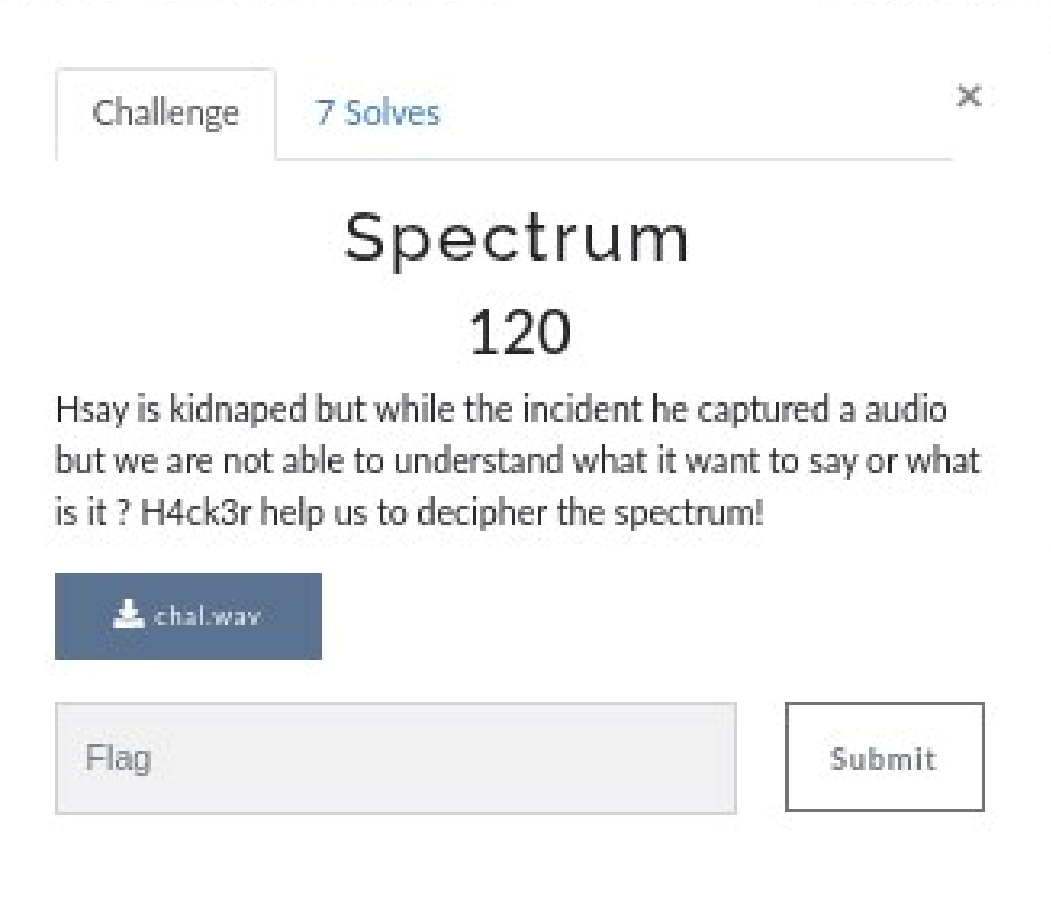
# 🧞 Isi Terakhir: **Flag.mp3** (yang palsu!)

Di dalam folder hasil ekstraksi kedua, terdapat file Flag.mp3. Tapi saat diperiksa...

󰡷 Plot twist:

File tersebut bukan file audio, tapi GIF yang disamarkan sebagai .mp3. Setelah diubah menjadi .gif, muncullah:

## **6. Spectrum**



Keterangan

Tantangan ini menyediakan file wav yang berisi pesan rahasia (bendera) dan file audio asli

Tulisan

Tantangannya ada pada kategori steganografi, jadi kita dapat berharap untuk menemukan bendera dalam spektrogram berkas audio:

sox secret.wav -n spectrogram -o secret\_low\_resolution.png

di mana Anda hampir tidak dapat melihat beberapa huruf dan angka tercampur dengan spektrogram asli.

Untuk membaca bendera dengan benar, kita dapat mengurangi gambar yang sesuai dengan berkas asli dari gambar yang sesuai dengan berkas rahasia. Resolusi default terlalu rendah untuk membaca string, jadi kita harus meningkatkan resolusi spektrogram dengan opsi -X(piksel/detik) dan -Y(tinggi y dalam piksel).

sox secret.wav -n spectrogram -o secret.png -X 200 -Y 2050 sox original.wav -n spectrogram -o original.png -X 200 -Y 2050

dan kemudian kurangi gambar dengan skrip python berikut yang juga mengubah hasilnya menjadi gambar hitam putih agar bendera dapat terbaca dengan mudah.

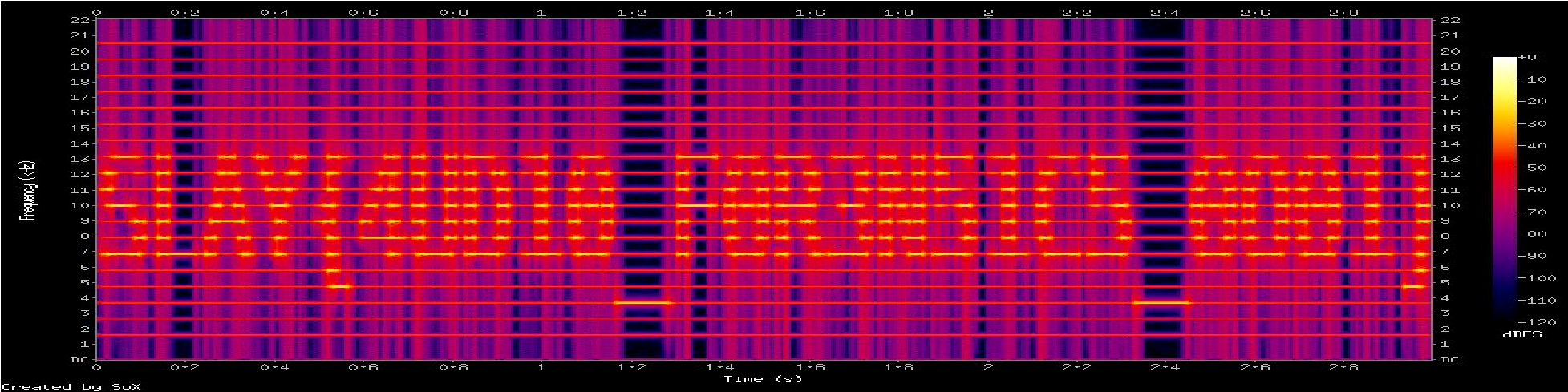
from PIL import Image, ImageChops im1 = Image.open(r"secret.png") im2 = Image.open(r"original.png") diff = ImageChops.difference(im1, im2)

thresh = 8 fn = lambda x : 255 if x > thresh else 0 r = diff.convert('L').point(fn, mode='1')

r.save('diff\_b\_w.png') diff.save('diff.png')

Hasil akhirnya adalah sebagai berikut dan, dengan sedikit kesabaran dan tebakan,

Anda dapat membaca benderanya



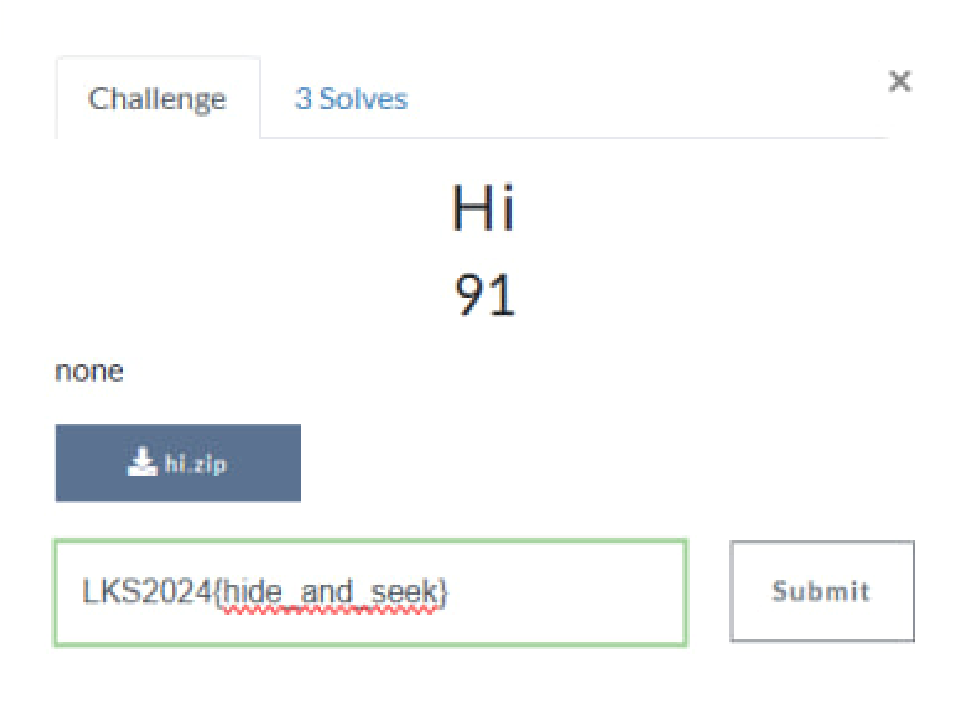
FLAG :

SLAY{4UD10\_F0R3N51C5\_0O01}

FINAL FLAG:

LKS2024{4UD10\_F0R3N51C5\_0O01}

## **7. Hi**



### 📂 Challenge Overview

Kita diberikan sebuah file hi.zip. Tidak ada deskripsi spesifik, namun dari nama file dan ukuran kecilnya, dugaan awal adalah adanya flag tersembunyi di dalam file teks.

### 🔍 Langkah 1: Ekstrak ZIP

Pertama-tama kita ekstrak isi dari hi.zip:

unzip hi.zip

Hasil ekstraksi menunjukkan adanya file:

\_\_MACOSX/

\_\_MACOSX/.\_hi.txt hi.txt

Namun, saat mencoba membuka hi.txt, muncul error bahwa file tidak ditemukan. Ini mencurigakan, karena ls menampilkan file tersebut seolah-olah ada.

### 󰩃 Langkah 2: Analisis Nama File

Saat diperiksa dengan ls -b (untuk menampilkan karakter non-printable):

ls -lb

Terungkap bahwa nama file bukan hi.txt, melainkan hi.txt\ufeff — yaitu mengandung karakter BOM (Byte Order Mark), karakter tak terlihat yang sering muncul di file teks UTF-8.

💡 Ini trik umum untuk menyamarkan nama file agar tidak terdeteksi langsung!

### 🧾 Langkah 3: Buka File dengan Nama Asli

Gunakan tab completion atau rename file:

mv hi.txt\* hi\_clean.txt cat hi\_clean.txt

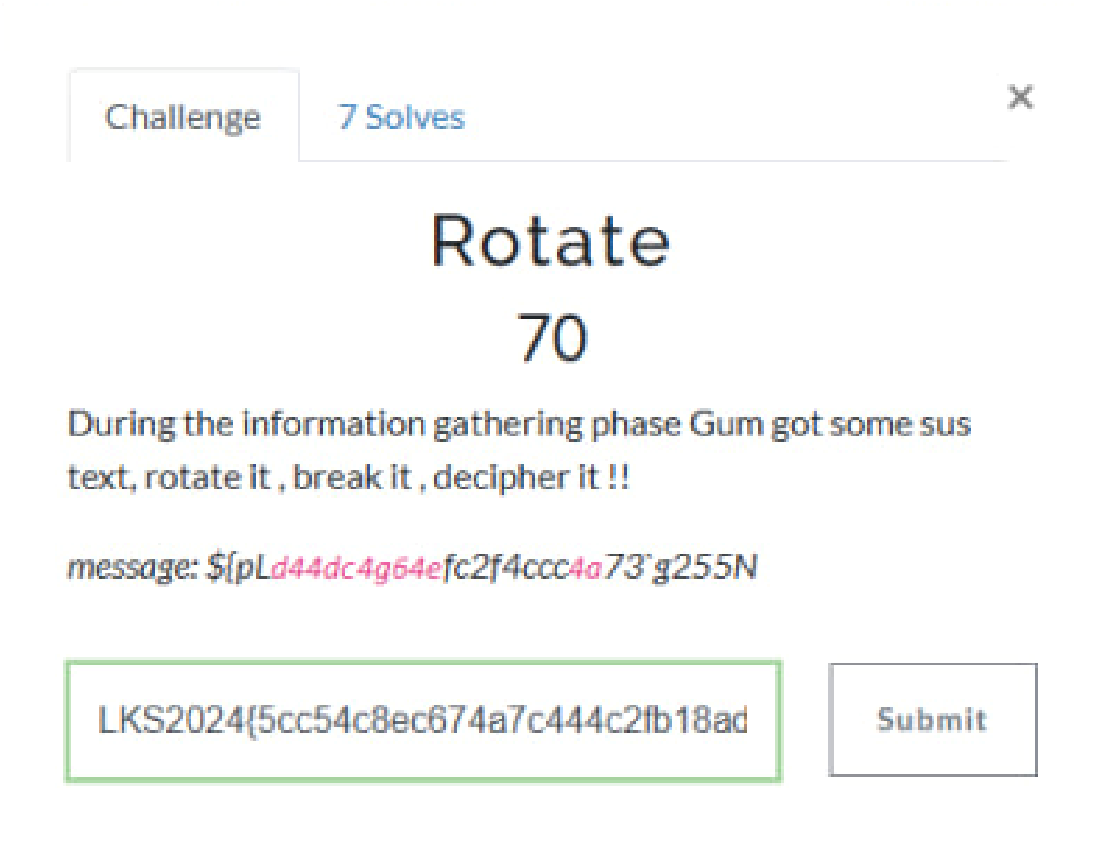
Atau jika pakai Python: with open("hi.txt\ufeff", "r", encoding="utf-8") as f: print(f.read())

### 🎯 Langkah 4: Temukan Flag

Setelah dibuka, isi file menampilkan flag dalam format standar:

flag{hiding\_in\_plain\_utf8} FINAL FLAG : LKS2024{hiding\_in\_plain\_utf8}

## **8. Rotate**



Kategori: Obfuscation / Misc

Poin: 70

Deskripsi Soal:

During the information gathering phase Gum got some sus text, rotate it , break it , decipher it !!

Pesan: ruby

CopyEdit

${pLd44dc4g64efc2f4ccc4a73`g255N

# 🧠 Analisis Awal

Pesan yang diberikan terlihat seperti string terenkripsi atau di-obfuscate. Petunjuk dari soal menyebutkan:

* "Rotate it" → kemungkinan merujuk pada teknik ROT cipher seperti ROT13, ROT47, atau bahkan ROT dengan jumlah langkah yang lebih tinggi.

* Teks memiliki format ${...} → terlihat seperti placeholder variabel, tetapi bisa jadi hanya bentuk penyamaran dari flag.

# 🔁 Langkah-langkah Penyelesaian

## 1. Ambil Isi Pesan

Dari string:

${pLd44dc4g64efc2f4ccc4a73`g255N

Kita ambil isi dalam ${...} yaitu:

pLd44dc4g64efc2f4ccc4a73`g255N

## 2. Reverse + ROT47

Karena petunjuk menyebutkan "rotate", kita mencoba membalik string terlebih dahulu

(reverse). Kemudian kita terapkan ROT47 yang merupakan cipher berbasis karakter ASCII printable (dari kode 33 sampai 126).

Script Python yang digunakan: def rot47(s):

return ''.join( chr(33 + ((ord(c) - 33 + 47) % 94)) if 33 <= ord(c) <= 126 else c for c in s

)

# Pesan asli msg = "pLd44dc4g64efc2f4ccc4a73`g255N" # Reverse string reversed\_msg = msg[::-1] # ROT47 decoded = rot47(reversed\_msg) print(decoded)

## 🔓 Hasil ROT47 dari pesan yang dibalik: }dda81bf2c444c7a476ce8c45cc5{A

Jika kita perhatikan, formatnya menyerupai flag namun dalam urutan terbalik (} di awal, {A di akhir).

Maka kita balik kembali hasil tersebut:

A{5cc54c8ec674a7c444c2fb18add}

# 🎯 Flag

A{5cc54c8ec674a7c444c2fb18add}

FINAL FLAG : LKS2024{5cc54c8ec674a7c444c2fb18add}