

Soal:

 "The 2048 Killer"

Para warga lokal melaporkan bau busuk dari sebuah rumah kosong, polisi terpaksa turun tangan. Mereka menemukan ember-ember berisi potongan daging di ruangan yang penuh tulisan "2048", beserta sebuah flash disk.

Khawatir aplikasinya berbahaya, mereka meminta bantuan kita untuk memecahkan masalah ini. Silahkan lihat foto dibawah.
<https://pusatkode.com/2048.png>

Diberikan sebuah url: <https://pusatkode.com/2048.png>

File tersebut merupakan sebuah gambar



Jika gambar tersebut dizoom pada bagian flashdisk, akan didapat sebuah url:
<https://pusatkode.com/2048.apk>



Setelah apk di decompile, saya mengetahui bahwa apk tersebut merupakan wrapper javascript. Sehingga inti utama aplikasi tersebut bukan java/kotlin, melainkan javascript yang berada di **assets\public\script.js**

Lempar script ke GenAI/LLM:

I got the URL: <https://gpousdcgce3cdx3ds6r32kwyga0igmsy.lambda-url.ap-southeast-1.on.aws/>

<https://gpousdcgce3cdx3ds6r32kwyga0igmsy.lambda-url.ap-southeast-1.on.aws/>

Ini seharusnya adalah url aws lambda function, sehingga hasilnya mungkin tidak statis. Saat saya mencoba fetch, hasilnya hanya 0 dan 1, dan setiap percobaan, hasilnya selalu berubah. Oleh karena itu saya coba ambil beberapa sample, lalu mencoba operator binary seperti OR, AND, dan XOR.

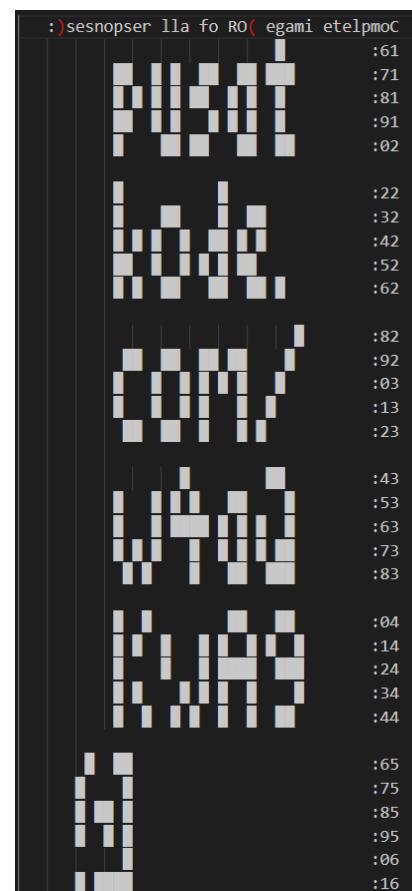
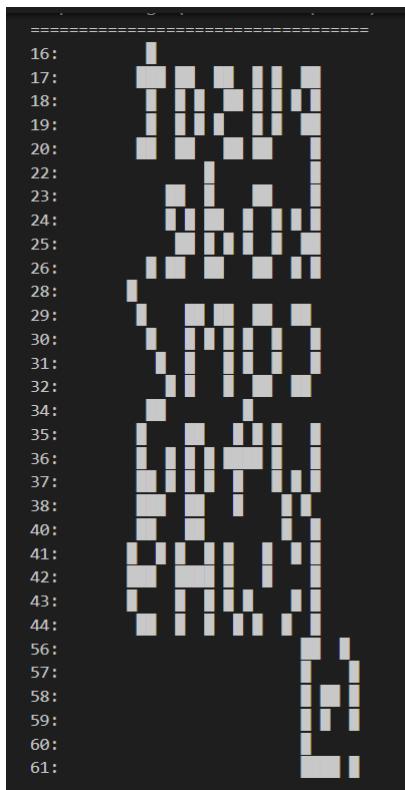
Hasil setelah operasi OR dan karakter diubah menjadi block agar readable, lalu split grid 32x64

Hanya OR

```
16: -----  
17: t02v  
18: 96o  
19: 3M0  
20: 6P4  
21: 8A9  
22:  
23:  
24:  
25:  
26:  
27:  
28:  
29:  
30:  
31:  
32:  
33:  
34:  
35:  
36:  
37:  
38:  
39:  
40:  
41:  
42:  
43:  
44:  
56:  
57:  
58:  
59:  
60:  
61:
```

Reverse

```
: )sesnopser lla fo RO( egami etelpmoC  
:61  
:71  
:81  
:91  
:02  
:22  
:32  
:42  
:52  
:62  
:82  
:92  
:03  
:13  
:23  
:43  
:53  
:63  
:73  
:83  
:04  
:14  
:24  
:34  
:44  
:65  
:75  
:85  
:95  
:06  
:16
```



Didapat url <https://pusatkode.com/w4aQKwA9>

Url tersebut merupakan file yang berisi html

Setelah dibuka di browser hanya akan menampilkan jam UTC. html DOMnya juga tidak ada script yang terrender

The screenshot shows the browser's developer tools with the DOM tree and the rendered page side-by-side.

DOM Tree:

```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  ... <body> flex
    <div id="clock">09:20:13</div>
    <div id="utc">UTC</div>
    <button id="button">REDIRECT</button>
  </body>
</html>
```

Renders:

09:20:14 UTC

The DOM tree shows the HTML structure with a `flex` class applied to the `body`. The rendered page displays the time "09:20:14" and the text "UTC".

Karena DOM diupdate setiap detik, maka kita hanya perlu break saat DOM diubah. Dapat dilakukan melalui devtools: klik kanan element html > break on > subtree modifications

Lalu script akan terpause dan terlihat

```
function updateClock() {
  const now = new Date();
  const hours = String(now.getUTCHours()).padStart(2, "0");
  const minutes = String(now.getUTCMinutes()).padStart(2, "0");
  const seconds = String(now.getUTCSeconds()).padStart(2, "0");
  const timeString = `${hours}:${minutes}:${seconds}`;
  const hmString = `${hours}:${minutes}`;

  document.getElementById("clock").textContent = timeString;

  // Check for specific times
  const triggerTimes = [
    "00:00",
    "00:02",
    "00:04",
    "00:08",
    "00:20",
    "00:22",
    "00:24",
    "00:28",
    "00:40",
    "00:42",
```

"00:44",
"00:48",
"02:00",
"02:02",
"02:04",
"02:08",
"02:20",
"02:22",
"02:24",
"02:28",
"02:40",
"02:42",
"02:44",
"02:48",
"04:00",
"04:02",
"04:04",
"04:08",
"04:20",
"04:22",
"04:24",
"04:28",
"04:40",
"04:42",
"04:44",
"04:48",
"08:00",
"08:02",
"08:04",
"08:08",
"08:20",
"08:22",
"08:24",
"08:28",
"08:40",
"08:42",
"08:44",
"08:48",
"20:00",
"20:02",
"20:04",
"20:08",
"20:20",
"20:22",
"20:24",
"20:28",
"20:40",
"20:42",
"20:44",
"20:48",
"22:00",
"22:02",

```

    "22:04",
    "22:08",
    "22:20",
    "22:22",
    "22:24",
    "22:28",
    "22:40",
    "22:42",
    "22:44",
    "22:48",
  ] ;

  if (triggerTimes.includes(hmString)) {
    callAPI();
  }
}

async function callAPI() {
  try {
    const response = await fetch(
      "https://oa6sjncccrkrswy57vw3744gm40ibuqo.lambda-url.ap-southeast-1.on.aws/"
    );
    const data = await response.text();
    const url = data;

    const button = document.getElementById("button");
    button.style.display = "block";
    button.onclick = () => window.open(url, "_blank");

    setTimeout(() => {
      if (button.parentNode) {
        button.remove();
      }
    }, 10000);
  } catch (error) {
    console.error("API call failed:", error);
  }
}

setInterval(updateClock, 1000);
updateClock(); // Initial call

```

Intinya, script akan memanggil API

<https://oa6sjncccrkrswy57vw3744gm40ibuqo.lambda-url.ap-southeast-1.on.aws/> pada jam-jam tertentu. Saat saya menemukan ini, outputnya hanya ?. Oleh karena itu saya membuat sebuah script untuk cek pada jam tersebut dan menyimpannya pada file.

Hasil dari lambda function tersebut pada jam 20:00 UTC atau 03:00 WIB
<https://pusatkode.com/UWQPprdi.txt>

Didapat sebuah txt file berisi unicode jika dilihat dengan hex editor (xxd)

Tanya ke LLM:

This CTF challenge involves a text file encoded with invisible/whitespace characters that mimics the structure of **Morse Code**.

1. Analysis of the Hex Dump

Looking at your `xxd` output, the file is composed of a repeating pattern of three specific byte sequences:

- **e3 85 a4**: This is the UTF-8 bytes for **U+3164 (HANGUL FILLER)**.
- **e2 a0 80**: This is the UTF-8 bytes for **U+2800 (BRAILLE PATTERN BLANK)**.
- **20**: This is the ASCII **Space**.
- **2e**: This is the ASCII **Dot/Period (.)**.

Decoded Output:

ADA BISIKAN DARI 2048 TUHAN, YANG MENJANJIKAN KEABADIAN DENGAN SYARAT TERTENTU. KONON DUNIA TIDAK SESEDERHANA YANG TERLIHAT, IA MEMILIKI 2048 DIMENSI, SEMENTARA MANUSIA TERKURUNG DI EMPAT YANG TERENDAH. UNTUK MENEMBUS LAPISAN YANG LEBIH TINGGI, DIPERLUKAN SEBUAH PENGORBANAN YANG DIPILIH DENGAN LOGIKA OBSESIF. SEORANG PEREMPUAN YANG SETIAP HARI BERDIRI DI DEPAN RUMAH MENJAJAKAN MINUMAN DINGIN DIANGGAP SEBAGAI VARIABEL YANG TAK DIJAGA, CUKUP UNTUK MEMBUKA JALAN MENURUT KEYAKINAN ITU. RITUALNYA DIGAMBARKAN DENGAN MEMOTONG TUBUHNYA SEBANYAK 2048 KALI, LALU DIKUBUR DI [PUSATKODE.COM/\(-6.2892995,106.9964141\)](https://pusatkode.com/(-6.2892995,106.9964141))

Url tersebut mengarahkan ke not found, namun karena saya menduga bahwa itu adalah koordinat (longitude, latitude), saya coba paste ke google maps.

Terdapat sebuah banner



Note: JANGAN DI TELFON/DOXING

Lalu saya mencoba url <https://pusatkode.com/081317155457>

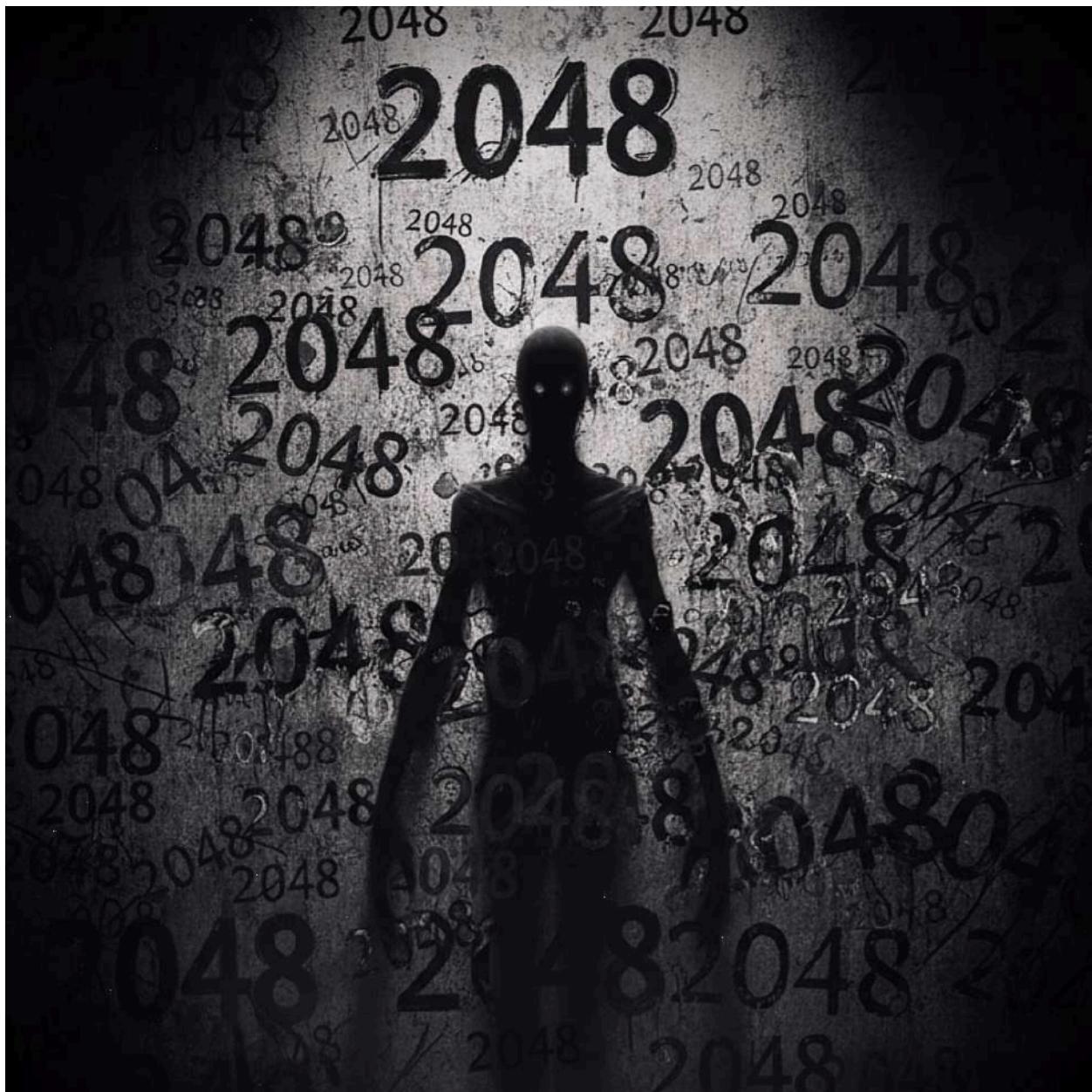
Terdapat sebuah video format .mov

Note: awas jumpscare



Didapat url <https://pusatkode.com/B6yadxhk.png>

Url tersebut berisi sebuah gambar



Pada gambar tersebut, terdapat 32 pixel alpha yang tidak solid (tidak 255 atau 0xff), hal tersebut membuat saya curiga dan mengecek value RGBA nya.

```
814,659: Alpha=212 RGBA=(104,104,104,212)
249,708: Alpha=96 RGBA=(116,116,116,96)
599,738: Alpha=38 RGBA=(100,100,100,38)
80,788: Alpha=200 RGBA=(116,116,116,200)
47,817: Alpha=42 RGBA=(110,110,110,42)
249,860: Alpha=36 RGBA=(32,32,32,36)
593,882: Alpha=74 RGBA=(54,54,54,74)
338,902: Alpha=48 RGBA=(115,115,115,48)
711,922: Alpha=57 RGBA=(56,56,56,57)
104,974: Alpha=172 RGBA=(105,105,105,172)
764,997: Alpha=75 RGBA=(121,121,121,75)
862,998: Alpha=168 RGBA=(112,112,112,168)
252,1069: Alpha=140 RGBA=(111,111,111,140)
rras002n49tsweaeba4htdtn 6s8iypo
```

Hal yang menarik disini adalah R=G=B. Coincidence? I think not!

Lalu angka tersebut berada di rentang 32-126, yang merupakan ASCII printable characters
Hasil convert ascii dan sorting: 00244689aaabdeehinnoprsssttwy, terdapat spasi, angka dan huruf, total 32 karakter.

Nah, angka angka disini bisa membentuk 2048, dan sisa 4 angka yang membentuk 4096. Lalu, huruf huruf tersebut dapat membentuk kata “password” dan tersisa `aabeehinnrstty`, karena stringnya sudah lumayan pendek saya coba pakai

<https://boulter.com/anagram/?letters=aabeehinnrstty>, copy all ke llm dengan prompt
Solve this anagram using words crafted using `aabeehinnrstty` and form sentence using additional known words `2048` and `4096`

Awalnya, saya mendapat “ANY 4096 IS BETTER THAN 2048”, tapi karena kalimatnya terasa aneh, mungkin sebenarnya kalimat lengkapnya `passwordnya 4096isbetterthan2048`. Spasi disini merupakan pemisah antara password asli dan label, dan menurut saya itu make sense. Namun setelah saya sumbit, password masih salah

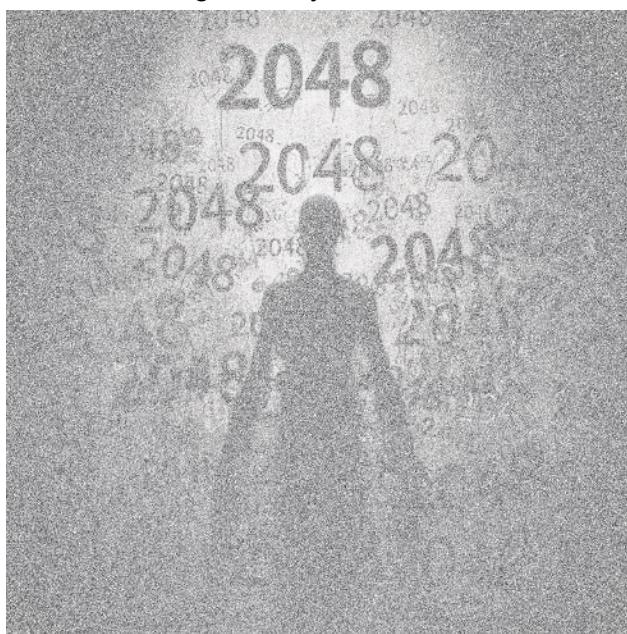
Setelah beberapa hari saya menunggu clue dari Vinco, cluenya adalah
Temukan sebuah nomor telepon (JGN DOX) di map, gunakan di jam spesifik

Lalu saya pun setup sebuah script untuk cek <https://pusatkode.com/081317155457> apakah berubah atau tidak. Tepat pada 20:48:00 WIB, videonya berubah, dan linknya menuju ke



<https://pusatkode.com/a0K3np1Y.png>

Seletah dicek, gambarnya identik 1080x1080, hanya berbeda di alpha sepertinya



Setelah compare setiap pixel di channel RGBA, ada 66 channel RGBA berbeda dari gambar sebelumnya

```
PS D:\Data.Project\ctf\vinco2> python .\check_image14.py
Comparing B6yadxhk.png and a0K3np1Y.png...
[+] Total non-zero values (all channels combined): 1166598
[!] Red channel differences: 66
[!] Green channel differences: 66
[!] Blue channel differences: 66
[!] Alpha channel differences: 1166400

[+] XOR image saved to: xor_result.png
[+] Amplified visual check saved to: xor_result_amplified.png
```

Alphanya hanya noise karena semuanya berbeda

Saat dicek posisi pixelnya, ternyata 32 pixel pada file sebelumnya merupakan subset (subhimpunan) dari 66 pixel ini.

Oleh karena itu saya hanya mengekstrak 34 pixel yang terekkslusi tadi.

```
PS D:\Data.Project\ctf\vinco2> python .\check_image24.py
114 114 114 4
111 111 111 8
98 98 98 244
100 100 100 40
112 112 112 224
97 97 97 242
97 97 97 80
```

Menariknya, pola R=G=B kembali ditemukan. Setelah diconvert ke ascii:

```
0248aaaaaaaaabdddeghikklnnoprrsstwy
```

Wah, anagram lagi. Dengan pola yang sama, eliminasi passwordnya, spasi, dan 2048, tersisa `aaaaaaabdeghikklnrt`, setelah mencari dengan tool anagram tadi <https://boulter.com/anagram/>, tidak ditemukan apa apa. Saya menduga ini adalah kalimat bahasa indonesia. Karena saya tidak menemukan anagram solver bahasa indonesia, saya meminta LLM untuk membuat script solvernya dengan backtracking dari dictionary kbki yang saya filter. Namun, kbki terlalu banyak bahasa daerah dan istilah istilah yang jarang digunakan dalam kehidupan sehari-hari.

SUCCESS: baketrak inha ala dag
SUCCESS: baketrak lada aan gih
SUCCESS: baketrak lada aga hin
SUCCESS: baketrak lada aga nih
SUCCESS: baketrak lada agi nah
SUCCESS: baketrak lada aha gin
SUCCESS: baketrak lada aha gni
SUCCESS: baketrak lada aha nig
SUCCESS: baketrak lada ahi ang
SUCCESS: baketrak lada ahi gan

Karena hasilnya terlalu banyak, lama dan banyak kalimat aneh, saya memutuskan untuk filtering manual dari 4710 kata bahasa indonesia yang sudah difilter berisi huruf 'aaaaaaabdeghikklnrt'. Rencananya, saya filtering kata yang jarang pakai excel, lalu masukkan dictionary yang lebih sedikit ke solvernya.

Saat sampai di kata 'adalah', saya berpikir itu bisa saja '2048 adalah...', namun saya belum menemukan ide untuk kata selanjutnya.

454	ang	x	
455	angad	x	
456	angah	x	
457	angai	x	
458	angair	x	
459	angalai	x	Sampai kata 'angka' di baris 468, saya berhenti lagi dan mencoba mengeliminasi huruf lainnya. Ternyata tebakan saya benar.
460	angalaki	x	
461	angat	x	
462	angel		
463	anget		
464	angi	x	
465	angidat	x	
466	angik	x	Sisa hurufnya 'abeikrt' dapat disusun menjadi 'terbaik'
467	angit	x	
468	angka		Sehingga hasilnya
469	angkai	x	
470	angkak	x	
471	angkar		
472	angkara		
473	angkat		
474	angkatid		
475	angke		
476	angkei		
477	angkek		
478	angkel		
479	angker		
480	angket		

passwordnya 2048adalahangkaterbaik