Writeups

ICC - de bluz

hiyeyi - Bryan Jericho

ztz - Muhamad Zibrisky

Pablu - Ady Ulil Amri

Writeups **1**
ICC - de bluz **1**
Cryptography **3**
Across the Tracks 3
Rahhh-SA 5
Web Exploitation **7**
Miku's Autograph 7
Grandma's Secret Recipe 10
Mary's Lamb is a Little Phreak 12
Beginner **14**
At Least It's Not Pandora 14
Break the Battalion 15
Inspector Requestor 16
Simon Says 17
Too Many Emojis 18
Straight Up Circular 19
Forensics **20**
QR Coded 20
Simon Says Scan This 21
Wordlands 23
Bucky's Impossible Obby 24
Logged 32
Miscellaneous **35**
World's Hardester Flag 35
Idea 1 36
Idea 2 37
Idea 3 37
It’s A Bird? 39
Mined Solving This? 41
OSINT **49**
April 25 49
Elite Stacker 50
Reverse Engineering **53**
Reversing for Ophidiophiles 53
theflagishere! 55
sus 59
Actual Reversing 61

Cryptography

Across the Tracks

10

whalker

I've been working on the railroad, all my live long day. We really should put up a fence, a deer just ran onto the tracks in a zig-zag pattern. After crossing my tenth track tracing the deer, I have found this message! What could it mean?

Samddre··ath·dhf@_oesoere·ebun·yhot·no··oso·i·a·lr1rcm·iS·aruf·toibadhn·nadpikud ynea{l_oeee·ch·oide·f·n·aoe·sae·aonbdhgo_so·rr.i·tYnl·s·tdot·xs·hdtyy'·.t·cfrlca·epeo· iufiyi.t·yaaf·.a.·ts··tn33}i·tvhr·.tooho···rlmwul·h·e·iHshonppsoleaseecrtudIdet.·n·BtIpd heiorcihr·or·ovl·c··i·acn·t·su··ootr·:b3cesslyedhelath·e·_

Soal ini menggunakan metode Rail Fence Cipher untuk menemukan flagnya, langsung saja dimasukkan di kodingannya

```
def rail_fence_decrypt(ciphertext, rails):
    fence = [['\n' for _ in range(len(ciphertext))] for _ in range(rails)]

    rail_pattern = list(range(rails)) + list(range(rails-2, 0, -1))

    index = 0
    for rail in rail_pattern * (len(ciphertext) // len(rail_pattern) + 1):
        if index < len(ciphertext):
            fence[rail][index] = '*'
            index += 1

    cipher_index = 0
    for r in range(rails):
        for c in range(len(ciphertext)):
            if fence[r][c] == '*' and cipher_index < len(ciphertext):
                fence[r][c] = ciphertext[cipher_index]
                cipher_index += 1

    result = []
    index = 0
```

```
    for rail in rail_pattern * (len(ciphertext) // len(rail_pattern) + 1):
        if index < len(ciphertext):
            result.append(fence[rail][index])
            index += 1

    return ''.join(result)

ciphertext =
"Samddre··ath·dhf@_oesoere·ebun·yhot·no··oso·i·a·lr1rcm·iS·aruf·toibadhn·n
adpikudynea{l_oeee·ch·oide·f·n·aoe·sae·aonbdhgo_so·rr.i·tYnl·s·tdot·xs·hdt
yy'·.t·cfrlca·epeo·iufiyi.t·yaaf·.a.·ts··tn33}i·tvhr·.tooho···rlmwuI·h·e·i
HshonppsoleaseecrtudIdet.·n·BtIpdheiorcihr·or·ovl·c··i·acn·t·su··ootr·:b3c
esslyedheIath·e·_"
rails = 10

decrypted_text = rail_fence_decrypt(ciphertext, rails)
print(decrypted_text)
```

Dan di terminal akan menghasilkan flag:

```
Some·ciphers·are·easier·to·solve.·Some·ciphers·are·harder·to·sol
ve.·You·definitely·could·brute·force·this·one·if·you·did·it·by·h
and.·I·had·to·do·that·recently·on·an·exam.·It·was·not·as·fun·as·
I·had·hoped.·But·that·is·okay.·I·hope·you·didn't·do·this·by·hand
.·Here·is·the·flag·tho:·bronco{r@1l_f3nc3_cip3rs_r_cool}
```

Flag : bronco{r@1l_f3nc3_cip3rs_r_cool}

Rahhh-SA

10

yoshie878

Behold! A modern take on an old crypto classic!

I call it RAHHH-SA! That's because with just a simple numerical inversion of RSA's rules, it's now unbreakable! RAHHH!

e = 65537
n = 3429719
c = [-53102, -3390264, -2864697, -3111409, -2002688, -2864697, -1695722, -1957072, -1821648, -1268305, -3362005, -712024, -1957072, -1821648, -1268305, -732380, -2002688, -967579, -271768, -3390264, -712024, -1821648, -3069724, -732380, -892709, -271768, -732380, -2062187, -271768, -292609, -1599740, -732380, -1268305, -712024, -271768, -1957072, -1821648, -3418677, -732380, -2002688, -1821648, -3069724, -271768, -3390264, -1847282, -2267004, -3362005, -1764589, -293906, -1607693]

Y'know what? I'm so confident in this new system I'll even share one of my deepest secrets!

p = -811

Ini merupakan soal kripto RSA, namun dalam chall ini ada suatu masalah :

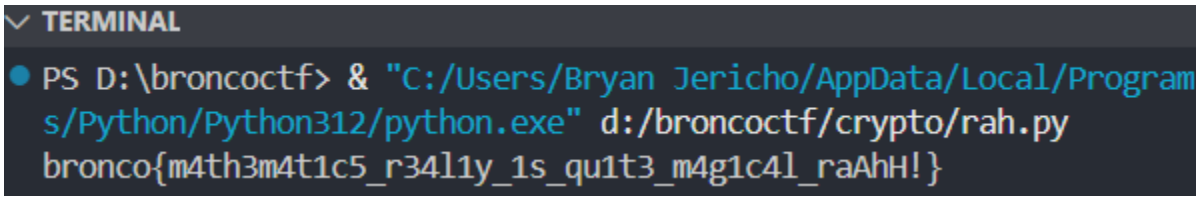
- Ciphertext yang diberikan bisa **bernilai negatif**.
- Kita perlu menangani cipher agar tetap valid dalam operasi modular.
- Cara mendekripsi tetap mengikuti langkah standar RSA dengan sedikit modifikasi.

Langsung saja kita masukkan di kodingan

```
from sympy import mod_inverse

def decrypt_rahhh_rsa(c, n, e, p):
    p = abs(p)  # Ensure p is positive
    q = n // p  # Finding q
    phi = (p - 1) * (q - 1)  # Calculating Euler's totient function
    d = mod_inverse(e, phi)  # Finding modular inverse of e mod phi

    decrypted_numbers = [pow((ci + n) % n, d, n) for ci in c]  # Handling
negative ciphertext correctly
    plaintext = ''.join(chr(num) for num in decrypted_numbers if 32 <= num
<= 126)  # Filter printable characters
    return plaintext
```

```python
# Given values
e = 65537
n = 3429719
p = -811
c = [-53102, -3390264, -2864697, -3111409, -2002688, -2864697, -1695722,
-1957072, -1821648, -1268305, -3362005, -712024, -1957072, -1821648,
-1268305, -732380, -2002688, -967579, -271768, -3390264, -712024,
-1821648, -3069724, -732380, -892709, -271768, -732380, -2062187, -271768,
-292609, -1599740, -732380, -1268305, -712024, -271768, -1957072,
-1821648, -3418677, -732380, -2002688, -1821648, -3069724, -271768,
-3390264, -1847282, -2267004, -3362005, -1764589, -293906, -1607693]

# Decrypt
flag = decrypt_rahhh_rsa(c, n, e, p)
print(flag)
```

Dan ketika dijalankan akan menghasilkan flag



Flag : bronco{m4th3m4t1c5_r34l1y_1s_qu1t3_m4g1c4l_raAhH!}

Web Exploitation

Miku's Autograph

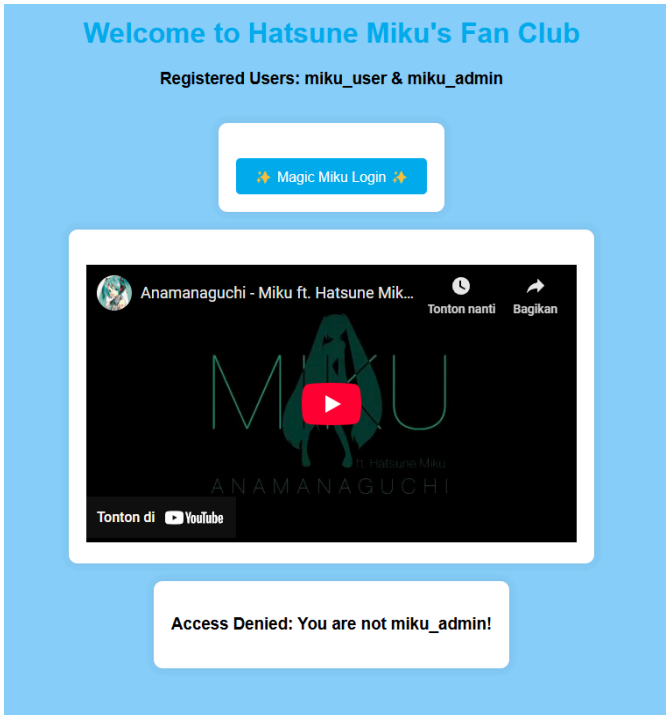
10

whalker

I am so proud of the fact that I have Miku's autograph. Ha! You don't!

https://miku.web.broncoctf.xyz

diberikan url sebuah website. saya mencoba untuk login namun gagal, “Access Denied: You are not miku_admin!“



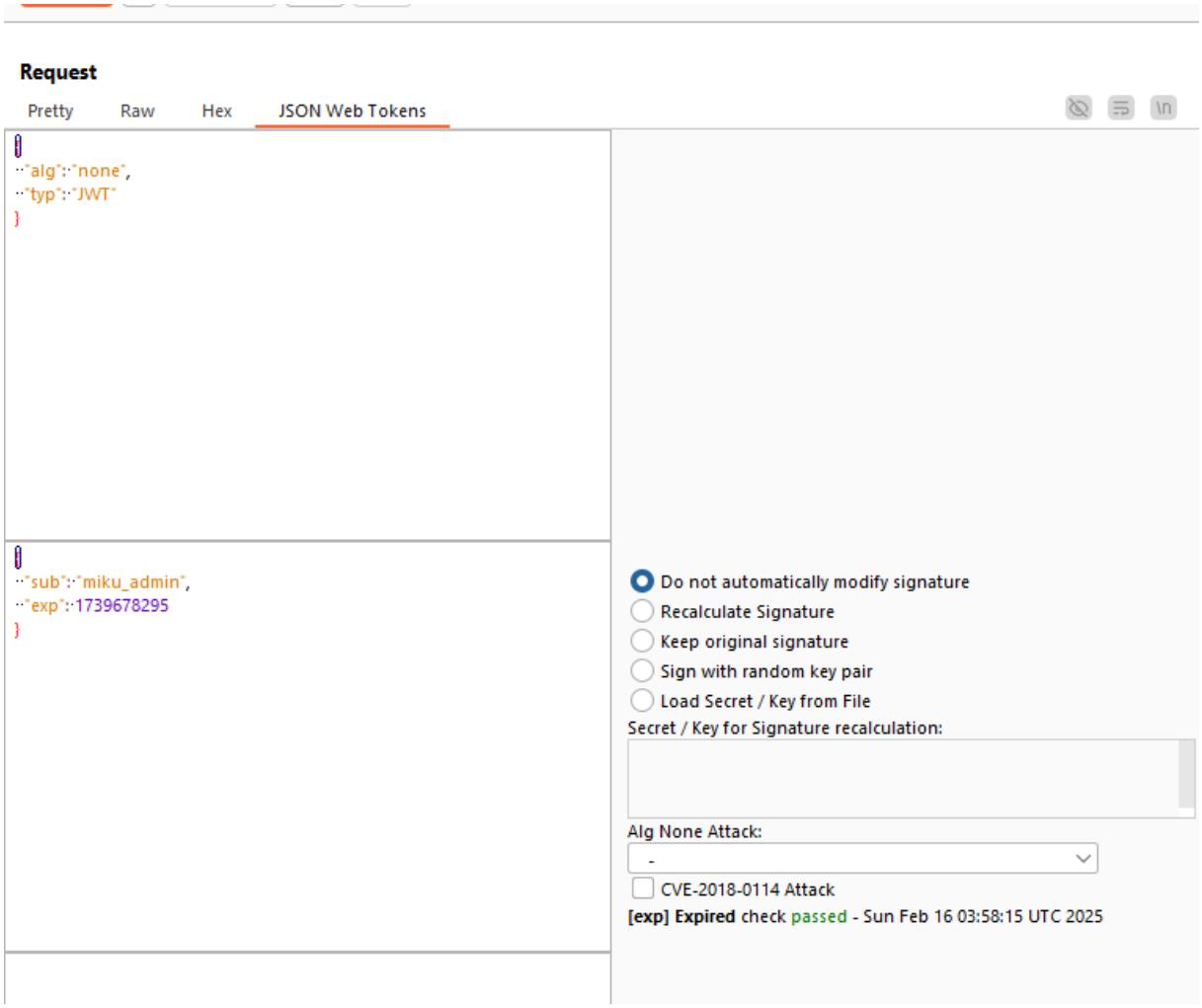
setelah saya cek source code pada bagian script sebagai berikut:

```
<script>
    function magicMikuLogin() {
        fetch('/get_token')
        .then(response => response.json())
        .then(data => {
            let token = data.your_token;
            fetch('/login', {
                method: 'POST',
                headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
                body: 'magic_token=' + encodeURIComponent(token)
            })
            .then(response => response.text())
            .then(result => document.body.innerHTML = result);
        });
    }
</script>
```

script ini tujuannya untuk membuat token. langsung saja saya kunjungi endpoint /get_token.


```
{"your_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJtaWt1X3VzZXIiLCJleHAiOjE3Mzk3ODY0NDR9.q4gN3buCAwd6Uov3sB95ja0o4l12m9lDNHxDq9vXHvc"}
```


lalu saya terpikir untuk menggunakan jwt none attack. dan mengganti datanya dari miku_user ke miku_admin



dan yap

```
<h2>
    Welcome, Miku Admin! Here's your flag: bronco{miku_miku_beaaaaaaaaaaaaaaaaam!}
</h2>
```

flag:
bronco{miku_miku_beaaaaaaaaaaaaaaaaam!}

Grandma's Secret Recipe

10

whalker

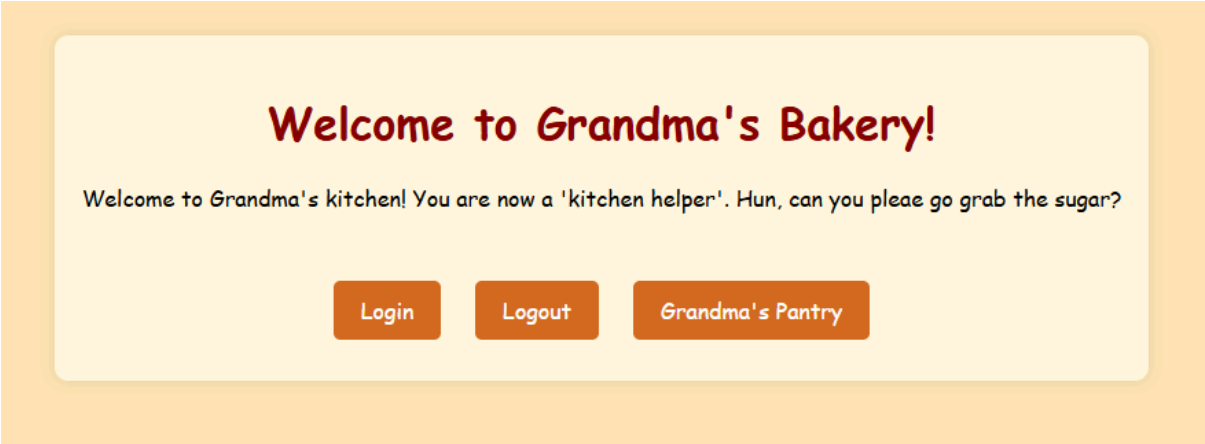
Grandma has been baking her world-famous cookies for decades, but she's always kept her secret recipe locked away. Nobody—not even her most trusted kitchen helpers—knows the full list of ingredients.

She insists it's all about "the perfect balance of love and a pinch of mystery", but deep down, you know there's more to it. Rumors say only Grandma herself is allowed to see the recipe, hidden somewhere in her kitchen.

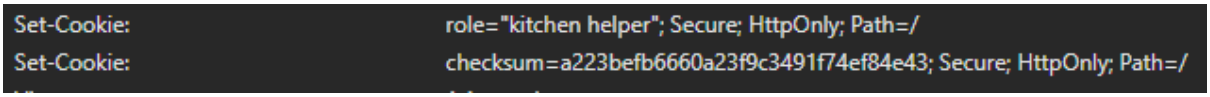
But, you were hired by Grandpa, who divorced her because she refused to share the recipe. Can you figure out the true secret behind her legendary cookies? 🍪👵

https://grandma.web.broncoctf.xyz

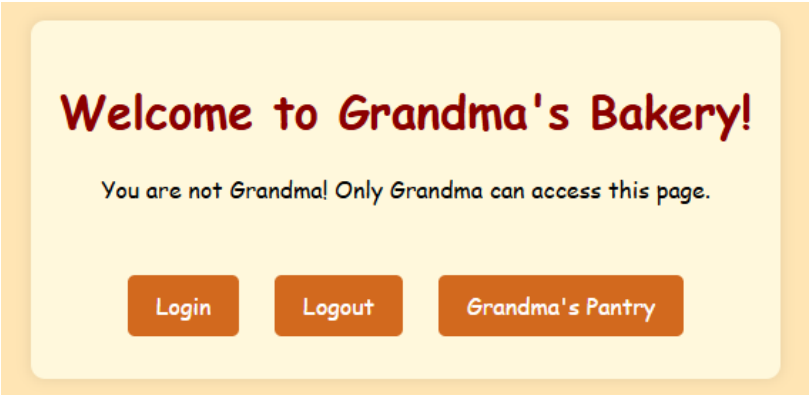
diberikan sebuah website, saya mencoba untuk login. dan berhasil.



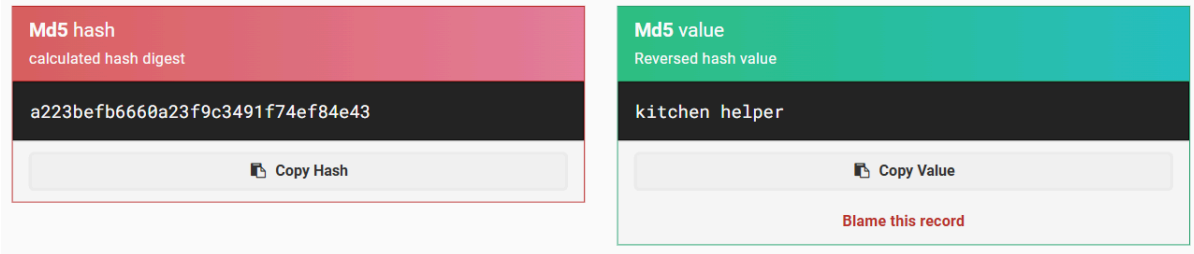
lalu ada cookie seperti ini:



saya mengklik Grandma's Pantry, namun kita harus login sebagai grandma.



ketika saya mereverse nilai hash dari checksumnya, ternyata nilainya merupakan role.



sehingga saya memutuskan untuk mengambil hashing md5 dari role : Grandma lalu men set sebagai cookie.

```
  Grandma&#39;s Secret Recipe:
</p>
<p class="flag">
  Flag: bronco{grandma-makes-b3tter-cookies-than-girl-scouts-and-i-wlll-fight-you-over-th@t-fact}
</p>
<br>
<a class="btn" href="/login">
```

Flag:
bronco{grandma-makes-b3tter-cookies-than-girl-scouts-and-i-will-fight-you-over-th@t-fact}

Mary's Lamb is a Little Phreak

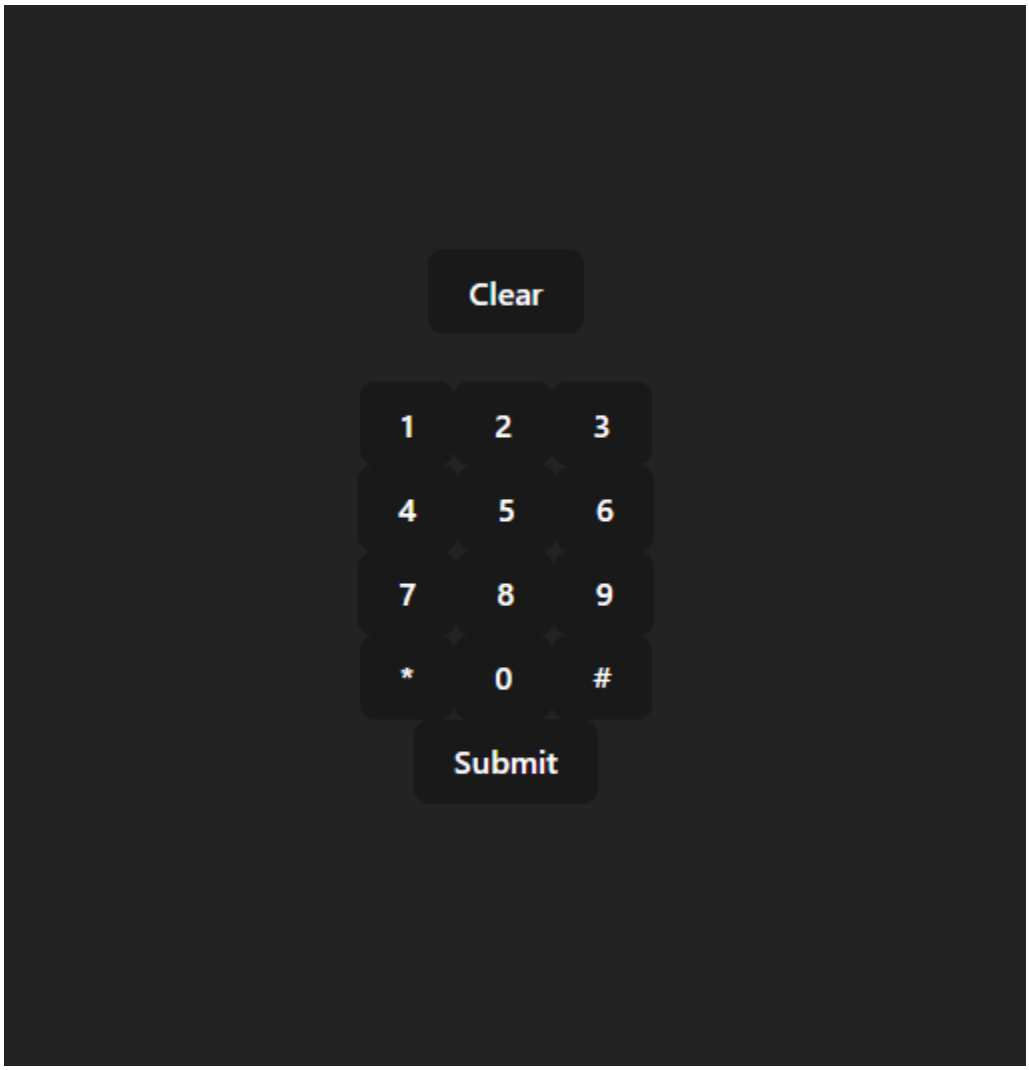
370

danny

I have this friend mary who has a lamb that only responds to a specific dial tone. Can you help mary find her lamb?

https://mary.web.broncoctf.xyz

diberikan website sbg berikut:



setelah ditekan tombolnya akan mengarahkan ke endpoind /mary/ . berdasarkan judul soal, yg dimana juga merupakan judul lagu “Mary's Lamb is a Little Phreak” sesuai deskripsi “only responds to a specific dial tone” jadi saya berfikir tone lagu tersebutlah yang direspon oleh web ini. Jadi saya mengirimkan request notes lagunya.

Request

Pretty Raw Hex

```
GET /mary/321233322239932123333222321 HTTP/2
Host: mary.web.broncoctf.xyz
Sec-Ch-Ua-Platform: "Windows"
Access-Control-Allow-Origin: *
Accept-Language: en-US,en;q=0.9
Accept: application/json, text/plain, */*
Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://mary.web.broncoctf.xyz/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

Response

Pretty Raw Hex Render

```
HTTP/2 200 OK
Server: Werkzeug/3.1.3 Python/3.13.2
Date: Sun, 16 Feb 2025 20:03:57 GMT
Content-Type: application/json
Content-Length: 55
Via: 1.1 google
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000

{
    "message":"bronco{WOah_yOu_fOund_m4rys_little_lamb}"
}
```

flag:
bronco{Woah_you_found_n4rys_little_lamb}

Beginner

At Least It's Not Pandora

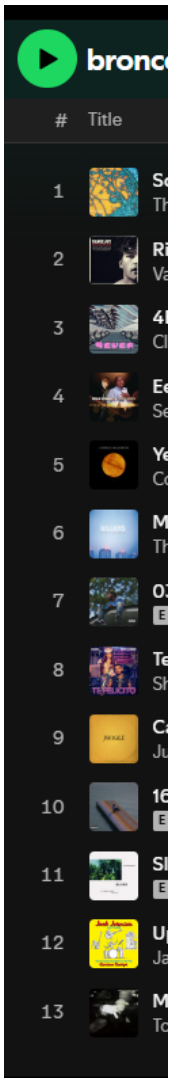
10

.tidalw

I really enjoy listening to music, but I hate that Spotify keeps shuffling my playlists. My taste in music used to be so backwards. P.S. One of the songs is my favorite song.

https://open.spotify.com/playlist/3UD6tVsCoVqal5BgXug19m?go=1&sp_cid=a0f9926371de38e180f302dedf1df658&nd=1&dlsi=3f0850ce261f4c27

Buka link nya, flagnya didapat karakter pertama tiap musik dari album bronco{} (dari terakhir ke awal)



flag:
bronco{MUS1CT0MY34RS}

Break the Battalion

10

tot_tater

You have received a file from the the infamous Bronco Battallion of the military. What is the correct input which gives you access to the military secrets? Format is bronco{input}.

File: a.out

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  char s1[264]; // [rsp+0h] [rbp-110h] BYREF
  unsigned __int64 v5; // [rsp+108h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  friendlyFunction(argc, argv, envp);
  puts("What is ze passcode monsieur?");
  __isoc99_scanf("%255s", s1);
  encrypt(s1);
  if ( !strcmp(s1, "brigade") )
    puts("correct password");
  else
    puts("wrong password");
  return 0;
}
```

```
int __fastcall encrypt(const char *a1)
{
  size_t i; // rax
  size_t v3; // [rsp+18h] [rbp-8h]

  v3 = 0LL;
  for ( i = strlen(a1); v3 < i; i = strlen(a1) )
  {
    a1[v3] ^= 0x50u;
    putchar(a1[v3++]);
  }
  return putchar(10);
}
```

“What is the correct input which gives you access to the military secrets?” untuk tahu apa input yang benar agar menghasilkan “brigade” tinggal xor dengan 0x50 atau memasukkan nya ke program.

```
┌──(pablu㉿d)-[~/sampah]
└─$ ./a
What is ze passcode monsieur?
brigade
2"97145
wrong password

┌──(pablu㉿d)-[~/sampah]
└─$ ./a
What is ze passcode monsieur?
2"97145
brigade
correct password
```

flag:
bronco{2"97145}

Inspector Requestor

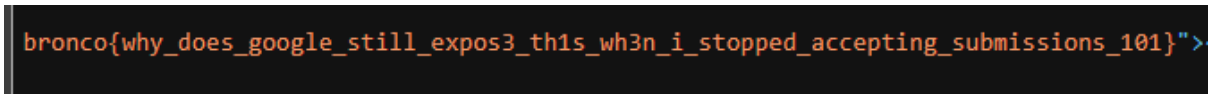
10

whalker

Want a flag? You need to request it via our Form, for efficiency.

https://forms.gle/oipmJZeVzYMrgKv39

cuma inspect doang.



flag:
bronco{why_does_google_still_expos3_th1s_wh3n_i_stopped_accepting_submissions_101}

Simon Says

10

tiffany_ttn

Help me play this game of simon says - remember, the last 2 lights have been blue!

File: simon.png

aperisolve > blue plane 2

flag:
bronco{simon_says_submit_this_flag}

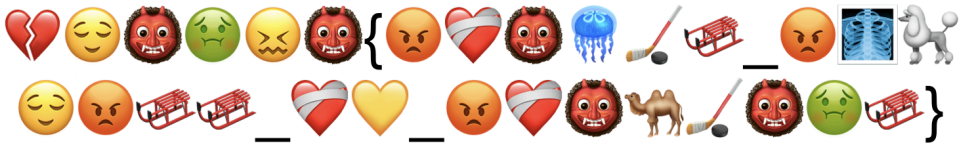
Too Many Emojis

10

tiffany_ttn

I like using emojis in my text messages, but my friend may have taken it too far. 💀
Can you figure out what she's trying to tell me? 🤔

Diberikan sebuah gambar seperti ini,



Ketika saya liat emojinya, saya tau kalau yang diambil ini hanya huruf depannya jadi saya cari berdasarkan nama emoji tersebut, refrensi : emojipedia.org

Jika digabungkan akan menjadi flagnya

Flag:
bronco{emojis_express_my_emotions}

Straight Up Circular

271

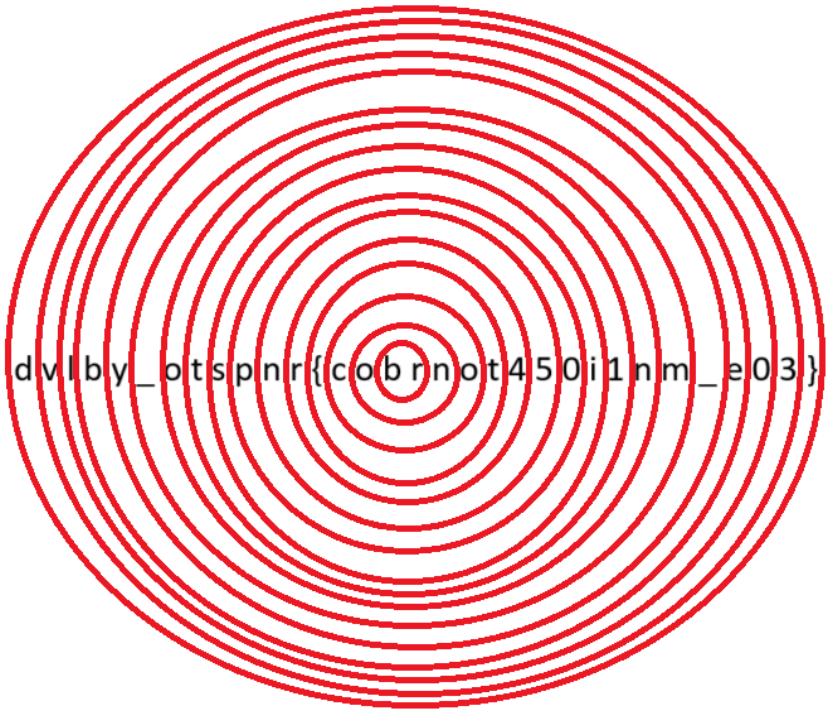
shwhale

Someone's been spreading weird messages around campus...

I keep seeing posters that have random characters on them, what could they mean?

Here's one of them: `dvlby_otspnr{cobrnot450i1nm_e03}`

So there is a hint in the title that it is related to Circular. After looking around and I found the pattern is like this:



Flag: `bronco{tr4n5p0sit1on_my_bel0v3d}`

Forensics

QR Coded

277

shwhale

This one should be really easy. All you have to do is scan a QR code!

File: easy_scan.png

stegsolve > gray bits

flag:
bronco{th1s_0n3_i5}

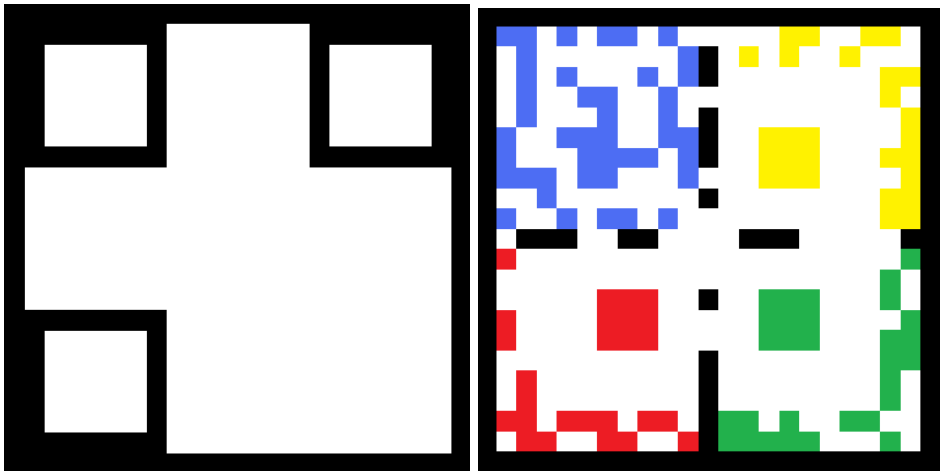
Simon Says Scan This

436

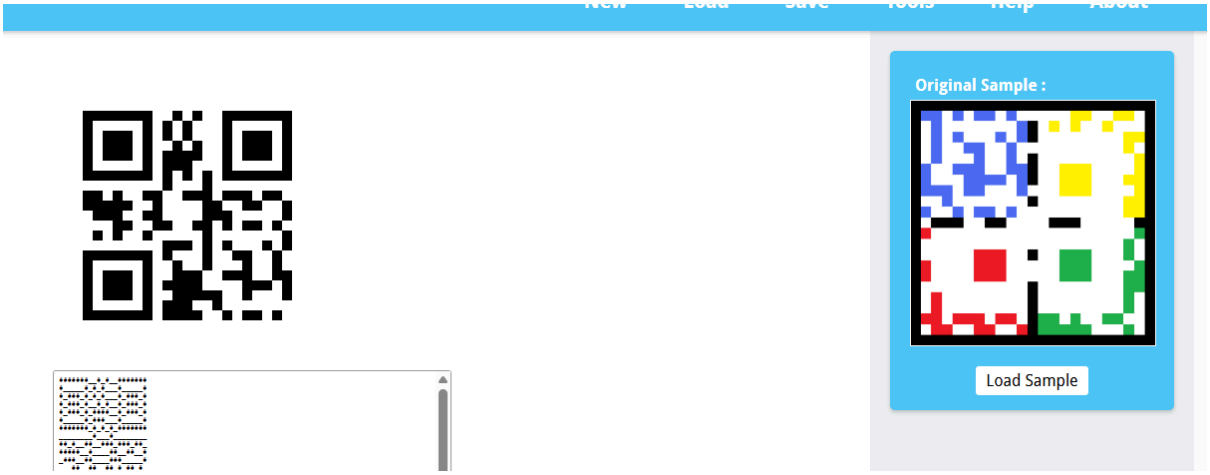
.tidalw

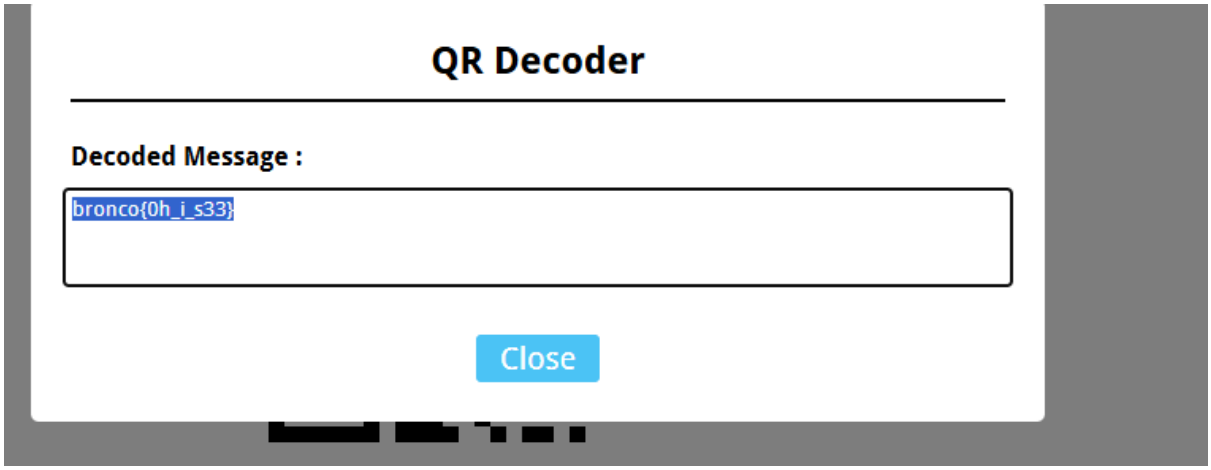
Ugh.. another QR Code challenge. I guess I better do what the title says.

Files: qr_code_p1.png, qr_code_p2.png



diberikan qr_code_p1 seperti struktur qr. lalu qr_code_p2 juga, namun secara keseluruhan, beserta empat bagian blue, yellow, green, dan red nampaknya juga semuanya di rotasi 180 derajat. jadi mari kita pulihkan qr nya. berikut qr yang telah dipulihkan. saya melakukannya secara manual, lebih gampang. saya melakukannya disini -> https://merri.cx/qrazybox/





flag:
bronco{0h_i_s33}

Wordlands

483

yoshie878

I keep getting advertisements for this thing called "Wordlands" wherever I go. I have seen their ads like 100 times this month and it's driving me crazy!

I decided to contact the creators of this mysterious application. However, they only replied with this strange image. "Only the worthy shall find the flag and cease the ads," they claim.

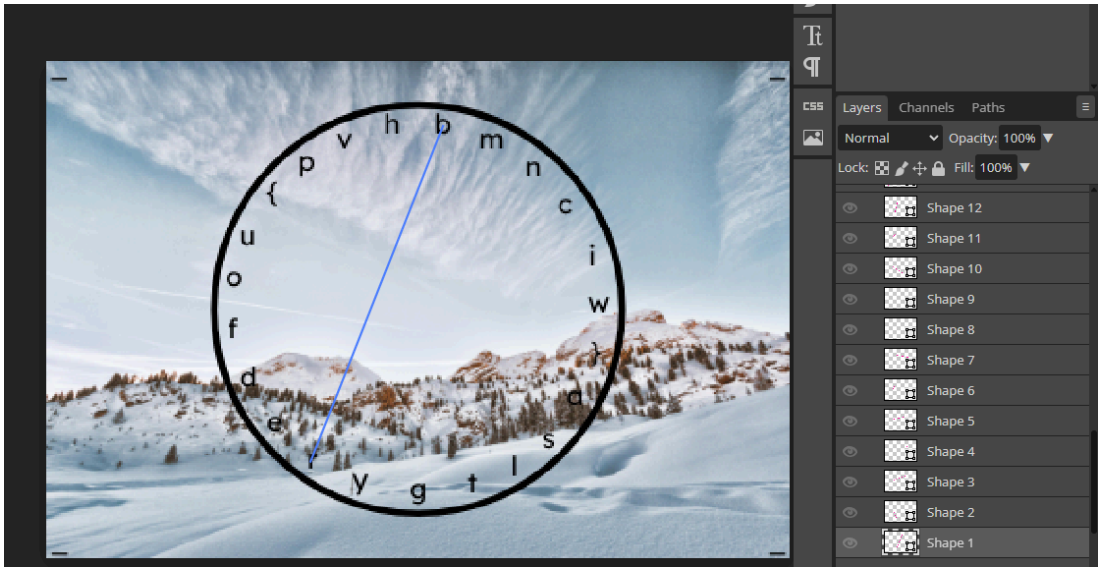
This is not funny!!! Help me uncover the secrets of the wordlands and make the ads stop!

File: wordlans

ketika saya zsteg, saya melihat sebuah file gambar Photoshop.

```
┌──(pablu㉿d)-[~/ctfluar/bronco/foren/wordlands]
└─$ zsteg wordlans.png
b1,r,lsb,xy         .. text: "`$aedJIMe\"am"
b1,g,lsb,xy         .. text: "b*\"J Bkc"
b1,b,lsb,xy         .. file: PGP symmetric key encrypted data - Plaintext or unencrypted data
b1,b,msb,xy         .. text: ["Z" repeated 17 times]
b1,rgb,lsb,xy       .. file: Adobe Photoshop Image, 400 x 267, RGB, 3x 8-bit channels
```

setelah diextract menggunakan command “zsteg -E b1,rgb,lsb,xy wordlans.png > out.ps” karna saya ga punya photoshop, untung ada https://www.photopea.com/ .



ikuti saja shape 1, shape 2, dst. dimana shape tersebut menghubungkan karakter-karakter tersebut yang membentuk sebuah flag.

flag:

bronco{i_love_admiring_beautiful_winter_landscapes}

Bucky's Impossible Obby

489

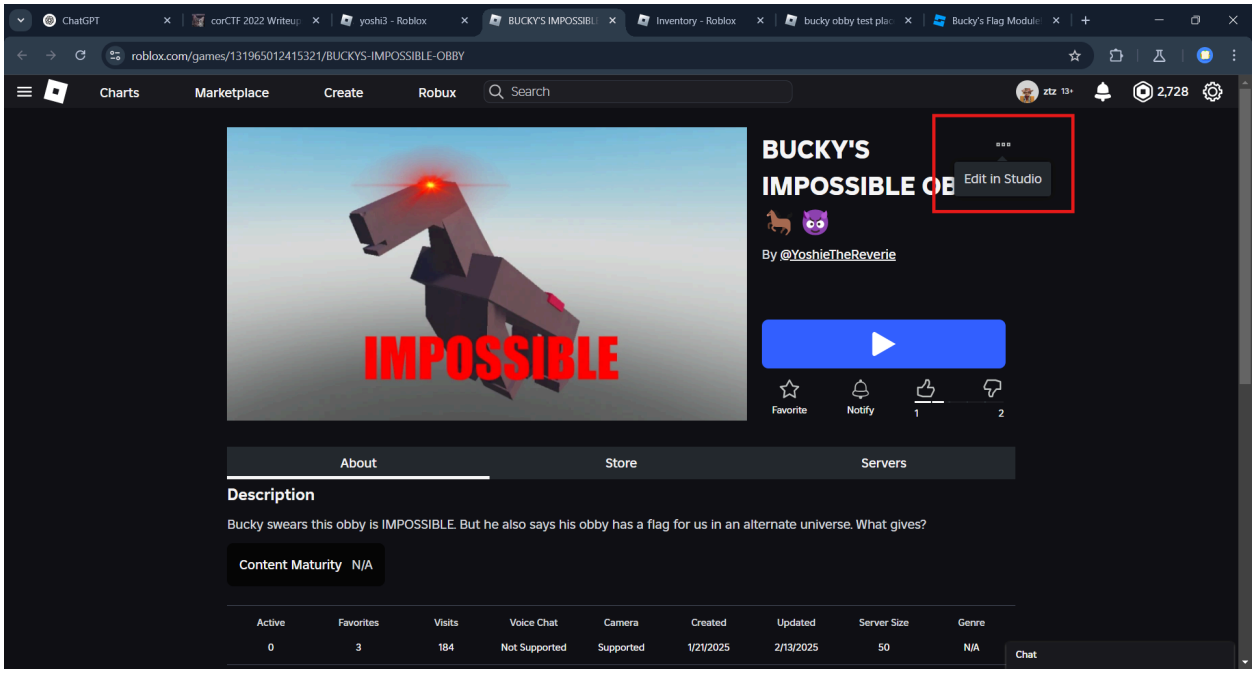
yoshie878

Bucky swears this obby (obstacle course) is IMPOSSIBLE. But he also says his obby has a flag for us in an alternate universe. What gives?
https://www.roblox.com/games/131965012415321/BUCKYS-IMPOSSIBLE-OBBY

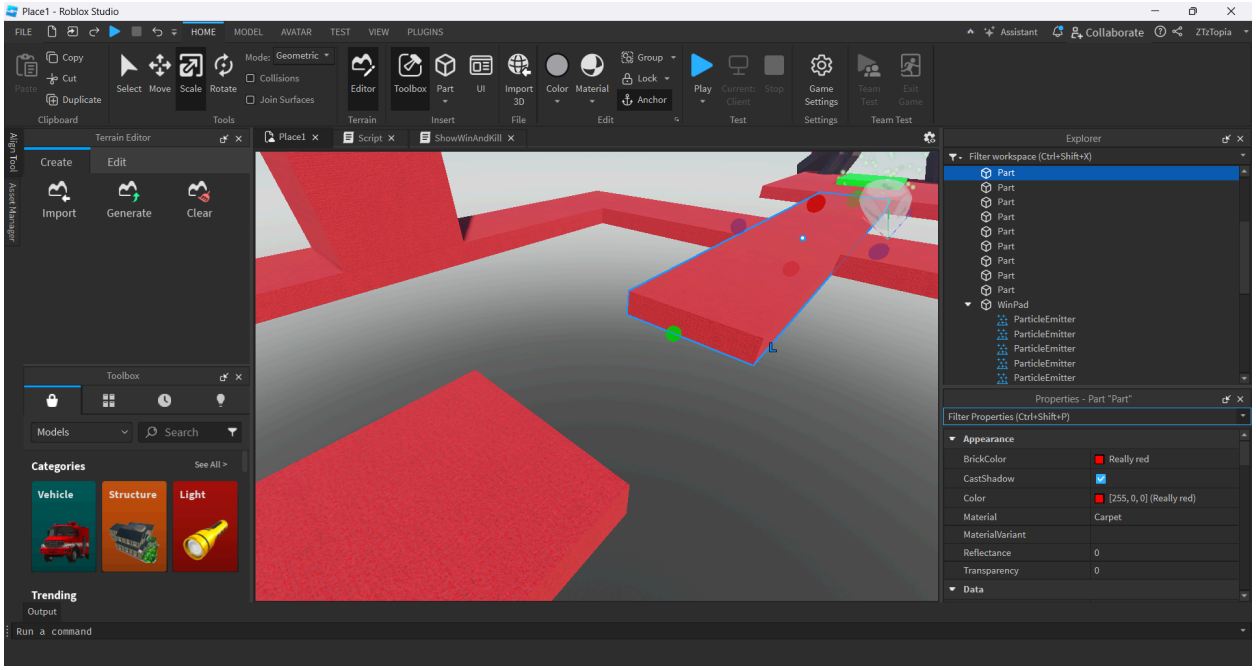
Note: Roblox Studio (and a Roblox account) is required for this challenge.

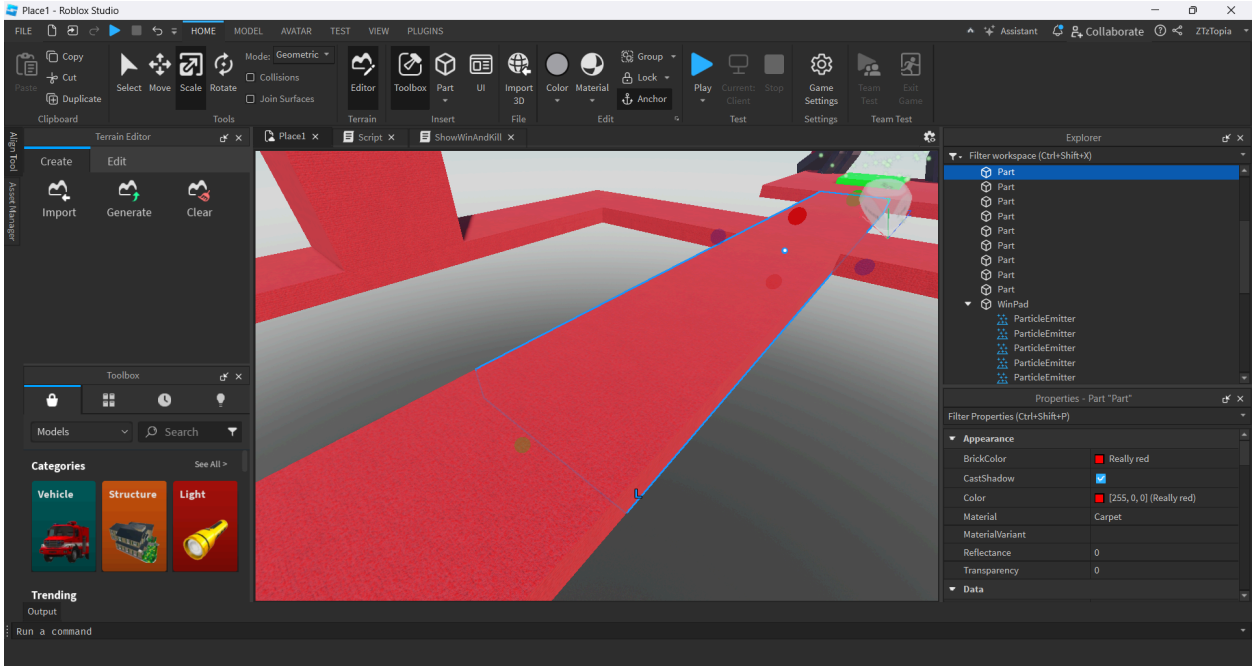
Yeay a **Robloks** challenge! I love playing with someone until midnight :)). There is a game called **BUCKY'S IMPOSSIBLE OBBY**. Where you won't be able to step on the finish checkpoint unless you use an exploit.

Because the note said **Roblox Studio** so I immediately went back to the game page and tried to see if there was an **Edit in Studio** option. Turns out there was!

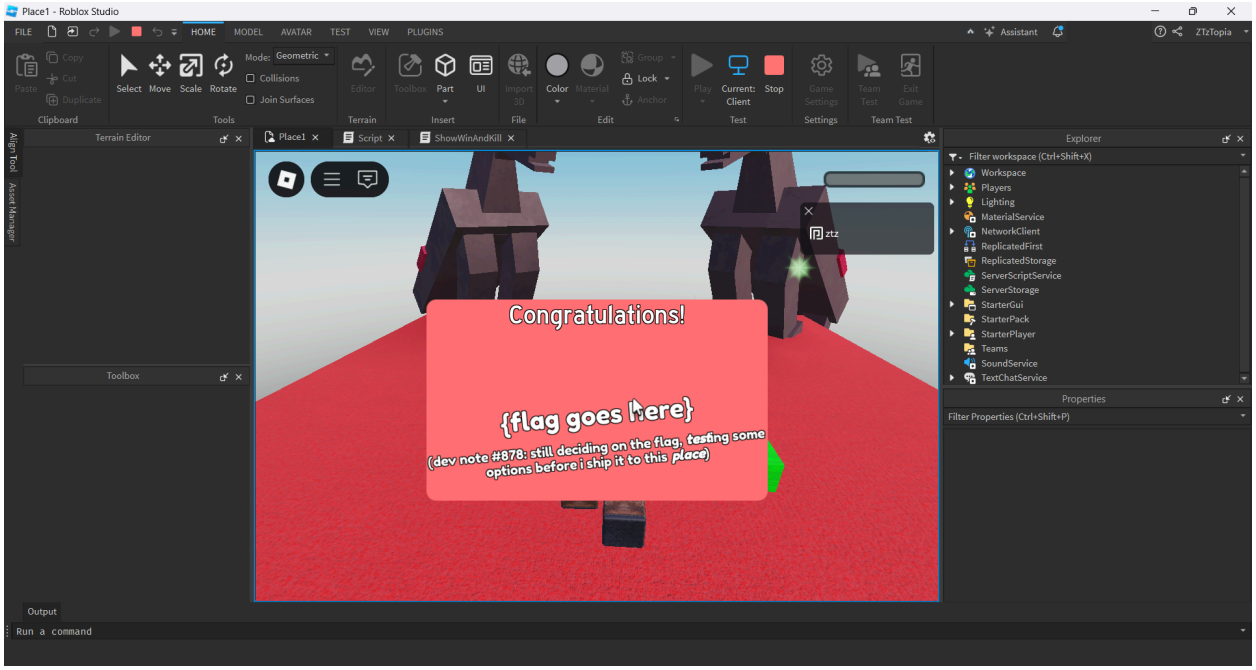


After successfully opening **Roblox Studio** I tried to make the path easier by scaling the object so that there are no holes and can go straight to the finish. After succeeding I tried to play the game and managed to get to the finish.

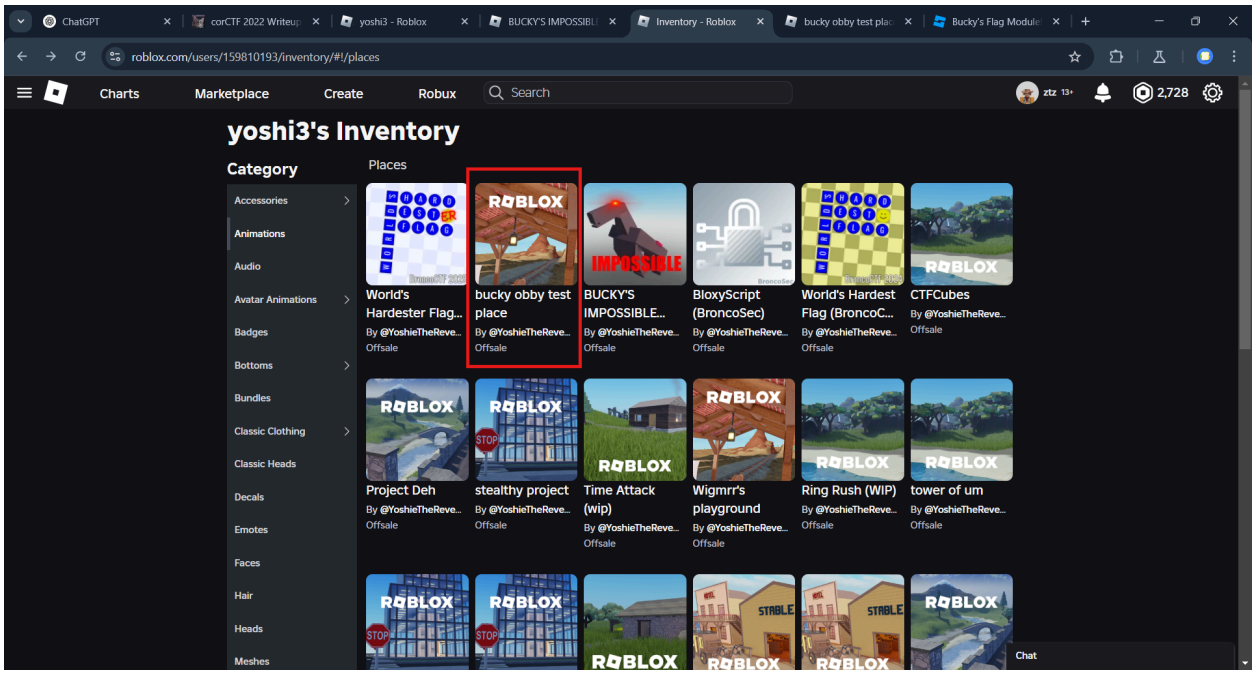




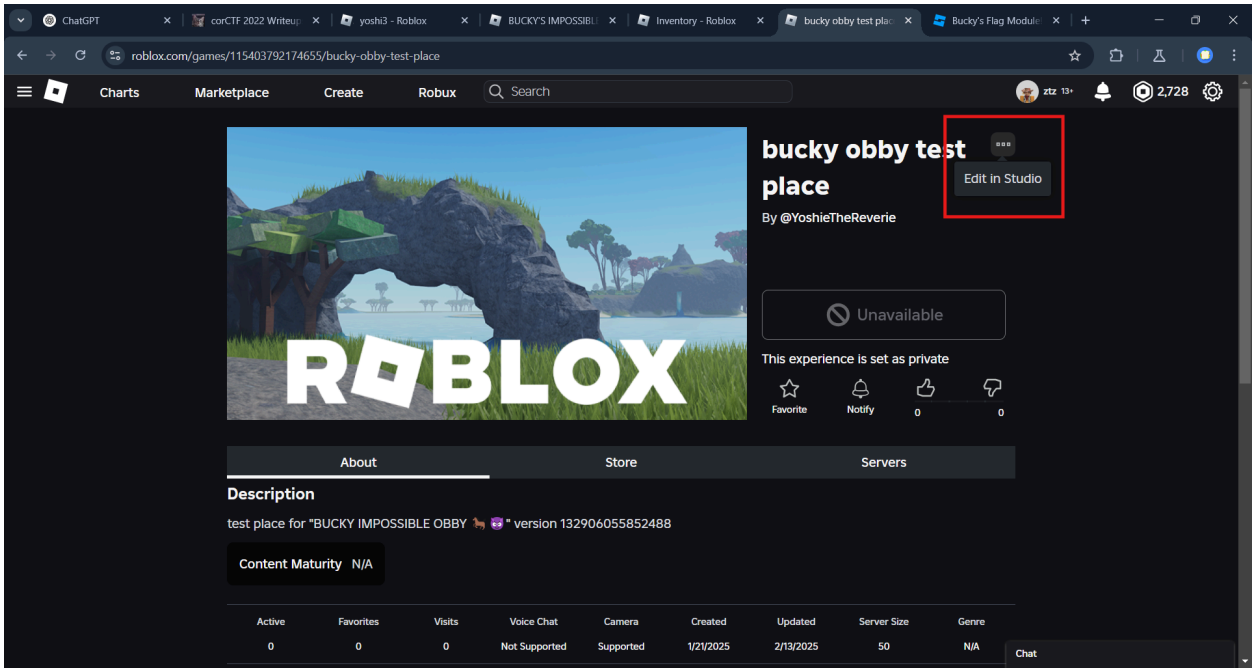
But when you get to the finish line, there's only a placeholder that says **{flag goes here}**.

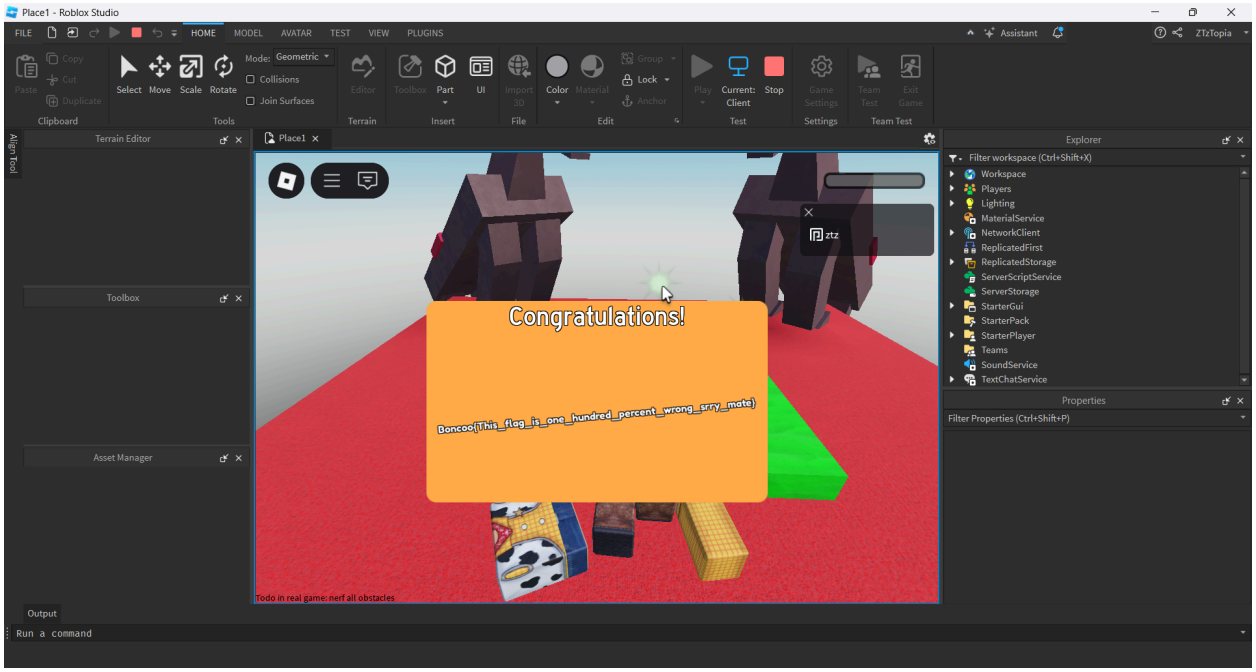


And there are dev notes which say: (dev note #878: still deciding on the flag, **test**ing some options before i ship it to this **place**) if you look closely there are the words **test** and **place** in bold. So let's just look for **test place** in the inventory of the game maker.

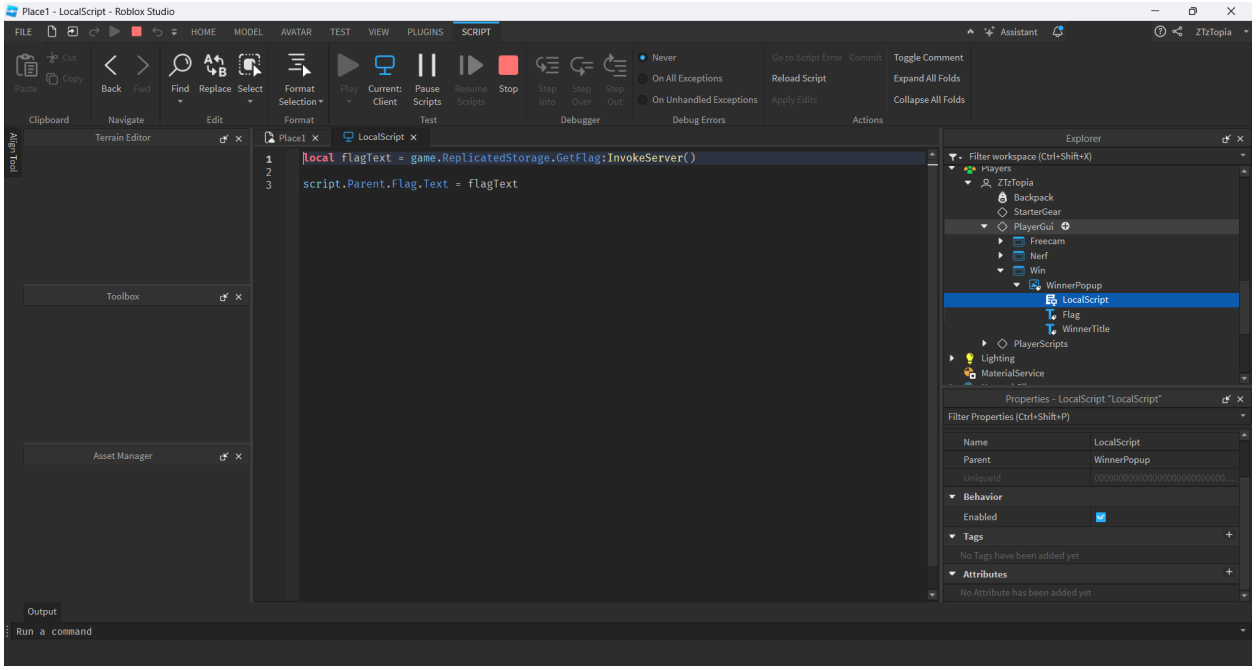


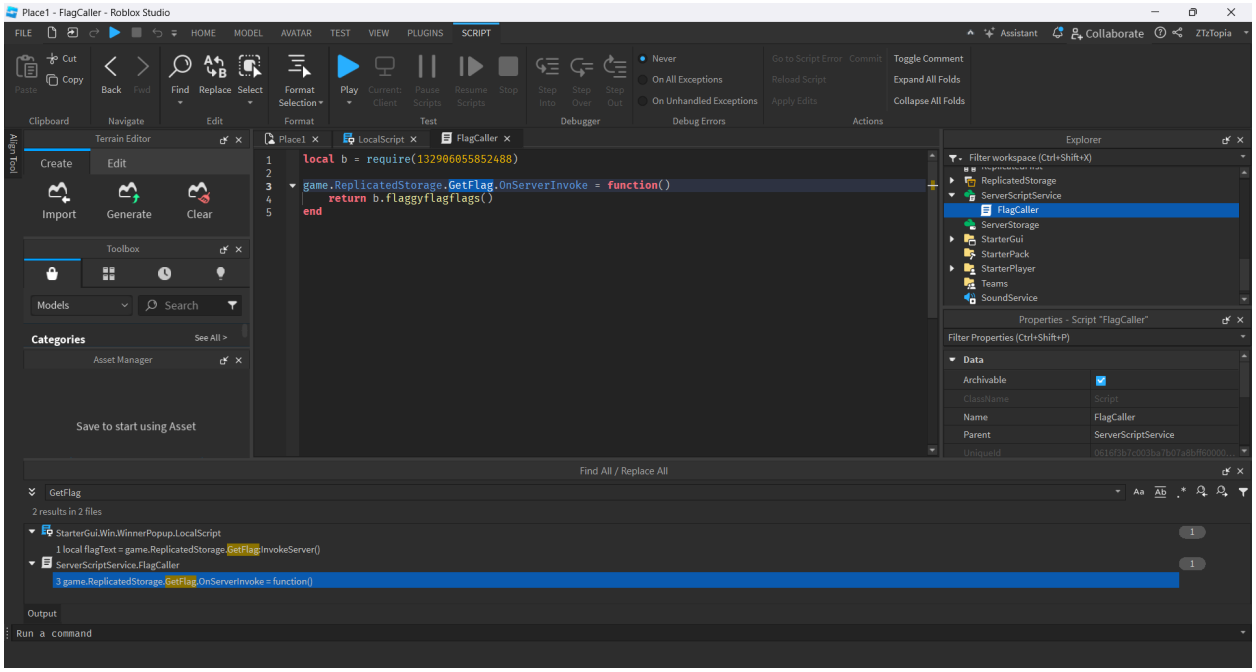
It turns out there is and it can be edited with **Roblox Studio**.



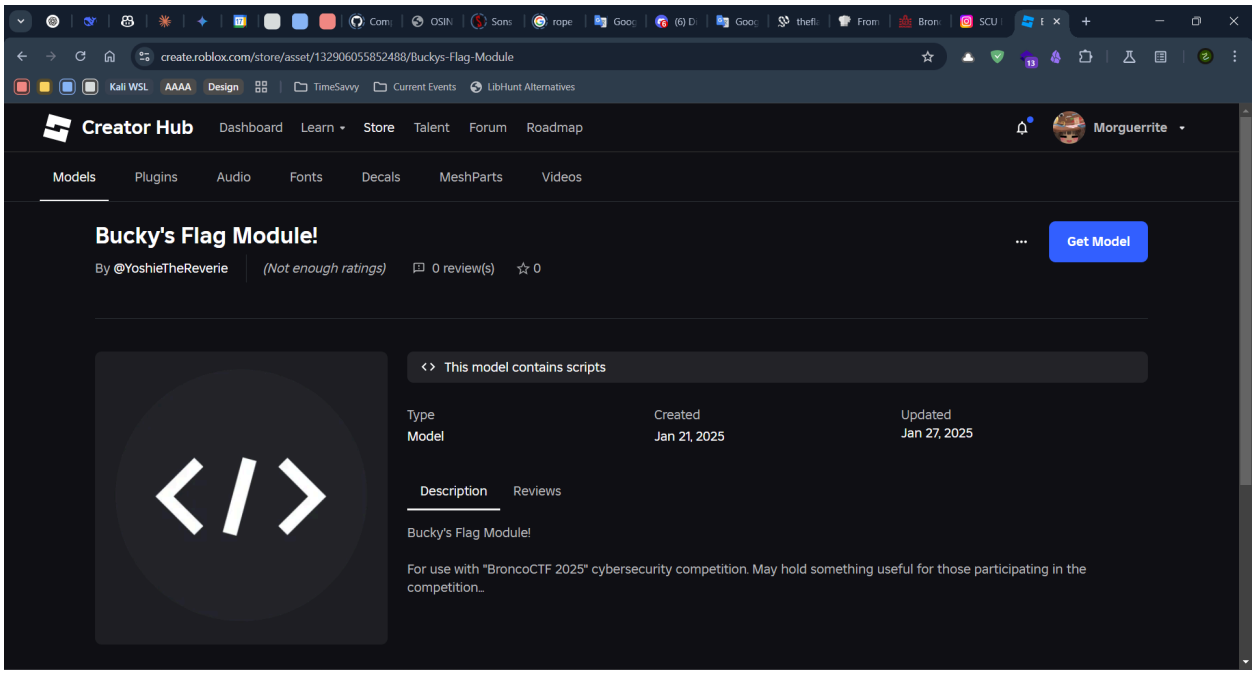


And another troll from the game maker. We can see the lua code used to create the game. It turns out that the flag that appears after finish is invoked from the server using `InvokeServer`.





We get that after the server receives the invoke from the client, it will return a flag that comes from a module. We can find that module by using the url `https://create.roblox.com/store/asset/<id>`.



After adding the module to our inventory we can see the lua code used to create the flag.

```lua
local buckysSuperAmazingFlagModule = {}
function buckysSuperAmazingFlagModule.flaggyflagflags()

    -- hey you!
    -- yeah you!
    -- stop snooping around!
    -- there's nothing to see here!

    -- ( - v - ) . z Z


    -- ( o - o )

    -- i'm his assistant!
    -- let's take a tour of this amazing module
    -- while he's asleep.

    -- important data!
    local data = {
        var_0107 = "_is_one_",
        var_0113 = "percent_",
        var_0103 = "his_flag",
        var_0011 = "11_w0w!}",
        var_0127 = "wrong_sr",
        var_0109 = "hundred_",
```

```
        var_0005 = "p0551bl3",
        var_0003 = "0t_s0_1m",
        var_0101 = "Boncoo{T",
        var_0002 = "bronco{n",
        var_0007 = "_4ft3r_4",
        var_0131 = "ry_mate}",
    }

    -- helper function
    local function ip(n)
        if n <= 1 then
            return false
        end
        for i = 2, math.sqrt(n) do
            if n % i == 0 then
                return false
            end
        end
        return true
    end
    -- helper function
    local ps = {}
    local n = 100
    local lim = 7
    while #ps < lim do
        if ip(n) then
            table.insert(ps, n)
        end
        n = n + 1
    end
    -- where the flag magic happens
    local flag = ""
    for i, p in pairs(ps) do
        flag = flag .. data["var_" .. string.format("%04d", tostring(p))]
    end

    -- and there you go!
    return flag
end
return buckysSuperAmazingFlagModule
```

By removing the variable used to troll us, we get the actual flag.

```
local data = {
    var_0011 = "11_w0w!}",
    var_0005 = "p0551bl3",
    var_0003 = "0t_s0_1m",
```

```
    var_0002 = "bronco{n",
    var_0007 = "_4ft3r_4",
}
```

We reorder the variables and get the actual flags.

```
local data = {
    var_0002 = "bronco{n",
    var_0003 = "0t_s0_1m",
    var_0005 = "p0551bl3",
    var_0007 = "_4ft3r_4",
    var_0011 = "11_w0w!}",
}
```

Flag: `bronco{n0t_s0_1mp0551bl3_4ft3r_411_w0w!}`

Logged

446

shwhale

We managed to get a keylogger on a notorious hacker's computer.

Our best analysts are pretty sure they type their password into a text file during this log segment, but we can't figure it out.

Can you get us the flag from this?

We are given a log file `keys.log` that contains keypresses and releases. The log file contains lines like the following:

```
KeyRelease event, serial 18, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36801639, (1314,474), root:(1322,554), state 0x0, keycode 50 (keysym 0xffe1),
same_screen YES
KeyPress event, serial 18, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0, time
36801639, (1314,474), root:(1322,554), state 0x0, keycode 50 (keysym 0xffe1),
same_screen YES
KeyRelease event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36801841, (1314,474), root:(1322,554), state 0x1, keycode 52 (keysym 0x3a),
same_screen YES
KeyPress event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0, time
36801841, (1314,474), root:(1322,554), state 0x1, keycode 52 (keysym 0x3a),
same_screen YES
KeyRelease event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36801908, (1314,474), root:(1322,554), state 0x1, keycode 50 (keysym 0xffe1),
same_screen YES
KeyRelease event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36801918, (1314,474), root:(1322,554), state 0x0, keycode 52 (keysym 0x3b),
same_screen YES
KeyRelease event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36802249, (1314,474), root:(1322,554), state 0x0, keycode 40 (keysym 0x65),
same_screen YES
KeyPress event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0, time
36802249, (1314,474), root:(1322,554), state 0x0, keycode 40 (keysym 0x65),
same_screen YES
```

```
KeyRelease event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36802332, (1314,474), root:(1322,554), state 0x0, keycode 40 (keysym 0x65),
same_screen YES
KeyRelease event, serial 21, synthetic NO, window 0x180000e, root 0x6c0, subw 0x0,
time 36802446, (1314,474), root:(1322,554), state 0x0, keycode 65 (keysym 0x20),
same_screen YES
```

The log file contains **keypresses** and **releases** for various keys. The `keysym` value is the `keysym` value of the key that was **pressed** or **released**. We can extract the `keysym` value from the log file and convert it to a character to get the plaintext.

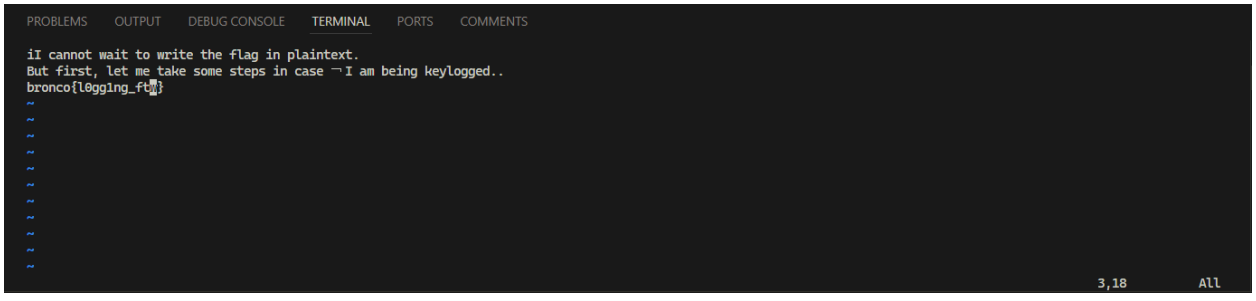
```python
import re

plaintext = []
shift_pressed = False

# Regex to extract keysym from the log line
keysym_regex = re.compile(r'keysym 0x([0-9a-fA-F]+)')

with open('keys.log', 'r') as file:
    for line in file:
        if 'KeyPress' in line or 'KeyRelease' in line:
            keysym_match = keysym_regex.search(line)
            if keysym_match:
                keysym = int(keysym_match.group(1), 16)

                if keysym == 0xffe1:
                    if 'KeyPress' in line:
                        shift_pressed = True
                    elif 'KeyRelease' in line:
                        shift_pressed = False
                elif keysym == 0xff0d:
                    if 'KeyPress' in line:
                        plaintext.append('\n')
                elif keysym == 0xff08:
                    if 'KeyPress' in line:
                        if plaintext:
                            plaintext.pop()
                elif keysym == 0xff1b:
                    if 'KeyPress' in line:
                        plaintext.append('<Esc>')
                else:
                    if 'KeyPress' in line:
                        # Convert keysym to character
                        try:
                            char = chr(keysym)
                            if shift_pressed:
```

```
                                if char.isalpha():
                                    char = char.upper()
                                else:
                                    plaintext.append('<Shift>')
                            plaintext.append(char)
                        except ValueError:
                            pass

print(f'Plaintext: {"".join(plaintext)}')
```

Running the script on the provided `keys.log` file gives us the following plaintext:

```
Plaintext: <Shift>:e flag.txt
iI cannot wait to write the flag in plaintext.
But first, let me take some steps in case ¬I am being
keylogged...<Esc>oasonethubkxxmloreucigqxlrcgmxlrcgmsneth xiexsuinthexsuebntixkxq
;aojeubs
m<Esc>Te;aro<Esc>lvhhhx0Pldtclrollla<Shift>{<Esc>wwa<Shift>}<Esc><Shift>%hxhxhs<Esc>ll
llr0hxlllrglR1n<Esc>lxllRft<Esc>llllrwhvTexllD<Shift>:s/m//g
3fga<Shift>_<Esc>twx¬:wq
→
```

The hacker tried to write flags in `flag.txt` but he tried to avoid **keylogger** by writing flags in a different way. Notice that some of the commands executed are `:e flag.txt` which means open the `flag.txt` file and `:s/m//g` which means delete all `m` characters from the file. So most likely this biggest hacker is using `vim` to write the flags. We can read the flags back in the same way as the hacker did.



Flag: `bronco{l0gg1ng_ftw}`

Miscellaneous

World's Hardester Flag

492

yoshie878

Hey. It's me, Mr. Dehnemy. Remember me from last year?

I have a new challenge for you. I present to you, the WORLD'S HARDEST(ER) FLAG!!

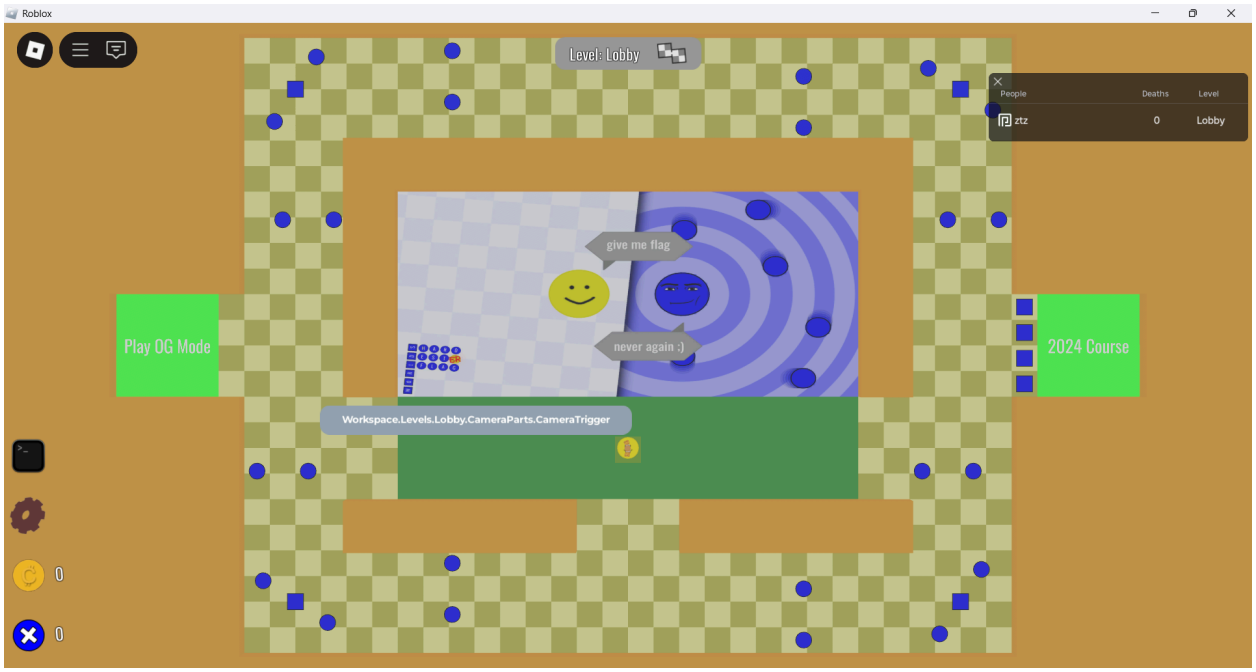
Your tactics from last year will not work. YOU WILL NOT GET MY FLAG!!!!!!!!!!!!!

https://www.roblox.com/games/97958089823595/Worlds-Hardester-Flag

Note 1: Roblox Player (and a Roblox account) is required for this challenge.

Note 2: If you load the game and get stuck on a blank colored screen, reset your character. Keep resetting your character if the screen keeps on being blank

The game is a **Roblox** game. The objective is to get the flag. There are two paths to get the flag. The first path is to go through old levels and the second path is to go through new levels. The flag is located at the end of the old or new levels.



Left is the new levels and right is the old levels. The flag is located at the end of the either new or old levels.

After looking around in the game, I found that we can execute lua scripts in the game.

I am familiar with **Lua** and **Roblox scripting** because I once made a script for a game for someone who often invited me to play roblox from morning to midnight WKKWKWK. So I tried to execute the lua script to get the flag.

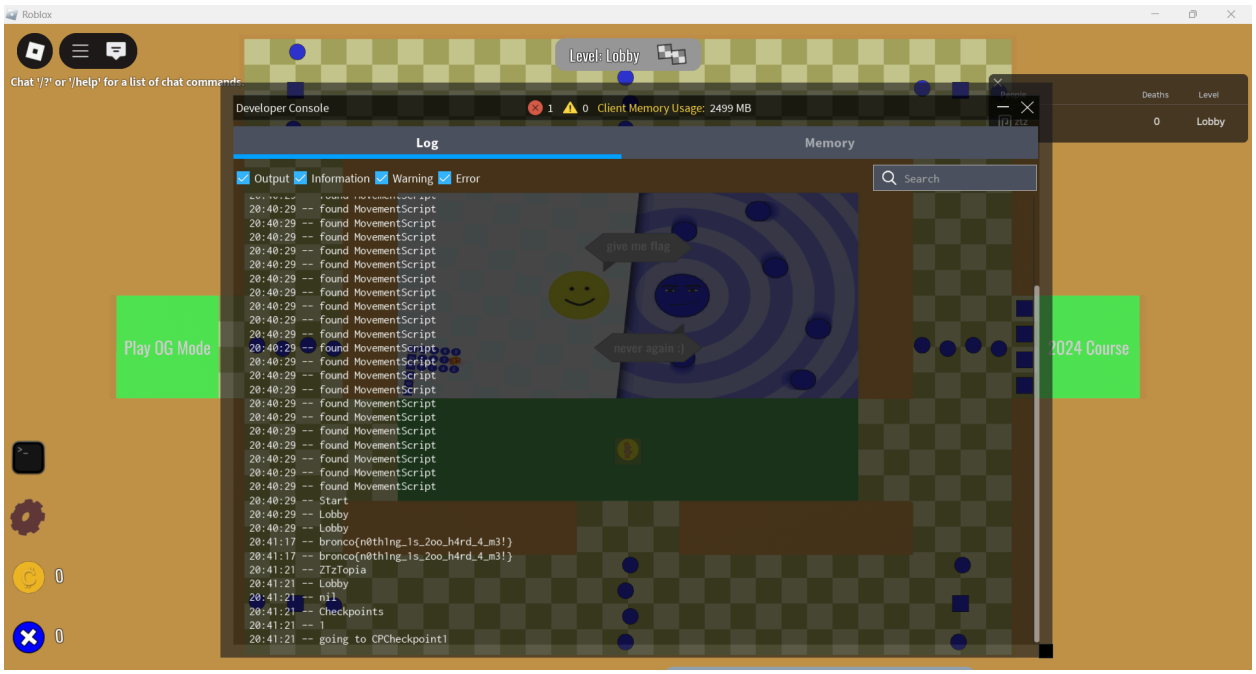
Idea 1

My first idea was to search for all the descendants of the game and look for an object that has the name `Flag`. It turns out that there is an object that has the name `Flag` after seeing the parent of the object is a `GUI` so there is likely to be a constant text on the object. Sure enough, it turns out that the object is a `TextLabel` containing the flag.

Because there are some banned words like `Name`, `Position` then we need to bypass them by using array and string concation.

```lua
for _, v in pairs(game:GetDescendants()) do
    if v["N" .. "ame"] == "Flag" then
        print(v.Text)
    end
end
```

Here we call `print` then the result will appear in the roblox console. We can see the roblox console by pressing `F9` or `CTRL + F9`.



Idea 2

Because I'm bored, I tried to be a hacker 😎, so I thought about being able to penetrate the blue object or obstacle by setting `CanCollide` to `false` or by clearing all children of an object by using the `ClearAllChildren` function.

Because in this game we are given the ability to debug objects in the game, we can use that information to remove obstacles in the game.

```
-- There is OG1, OG2, OG3, OG4 (final level)
Workspace.Levels.OG1.Hazards:ClearAllChildren()
```

After removing the obstacles in the game, we can easily pass all the levels in the game.

Idea 3

Directly teleport to the last checkpoint and get the flag. There are two checkpoints we can teleport to, namely `WinPad` at the `OG4` level and `OldWinPad` at the `Demo` level.

```
-- Workspace.Levels.OG4.Checkpoints.WinPad
-- Workspace.Levels.Demo.Checkpoints.OldWinPad
```

```lua
local localPlayer = game.Players.LocalPlayer
local character = localPlayer.Character or localPlayer.CharacterAdded:Wait()
-- local manusia = character:WaitForChild("H" .. "umanoid")

character:PivotTo(CFrame.new(Workspace.Levels.OG4.Checkpoints.WinPad["P" ..
"osition"]) + Vector3.new(0, 5, 0))
```

Flag: `bronco{n0th1ng_1s_2oo_h4rd_4_m3!}`

It’s A Bird?

448

whalker

I got gifted a raven. I didn't want the raven, but it is mine now. I am going insane, I swear, it just keeps yapping!

My friend, an ornithophile, swears it is trying to tell me something. That there is a deeper message, hidden within. But I dunno, man. It just keeps squawking.

steghide extract -sf myBirb.jpg --passphrase ""

Then take the bird.csv there are ascii number embeded within it

```
└─$ python
Python 3.12.8 (main, Feb  1 2025, 21:32:22) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> chr(98)
'b'
>>> chr(114)
'r'
>>> chr(111)
'o'
>>> chr(110)
'n'
>>> chr(99)
'c'
>>> chr(111)
'o'
>>> chr(123)
'{'
>>> chr(105)
'i'
>>> chr(60)
'<'
>>> chr(51)
'3'
>>> chr(112)
'p'
>>> chr(108)
'l'
>>> chr(97)
```

'a'
>>> chr(110)
'n'
>>> chr(101)
'e'
>>> chr(115)
's'
>>> chr(125)
'}'
example
MSG,6,333,3,A57B4D,103,2025/02/12,23:38:49.983,2025/02/12,23:38:49.983,,15800,,,,,0098,
0,0,0,0
0098 -> 98

Flag:
bronco{i<3planes}

Mined Solving This?

494

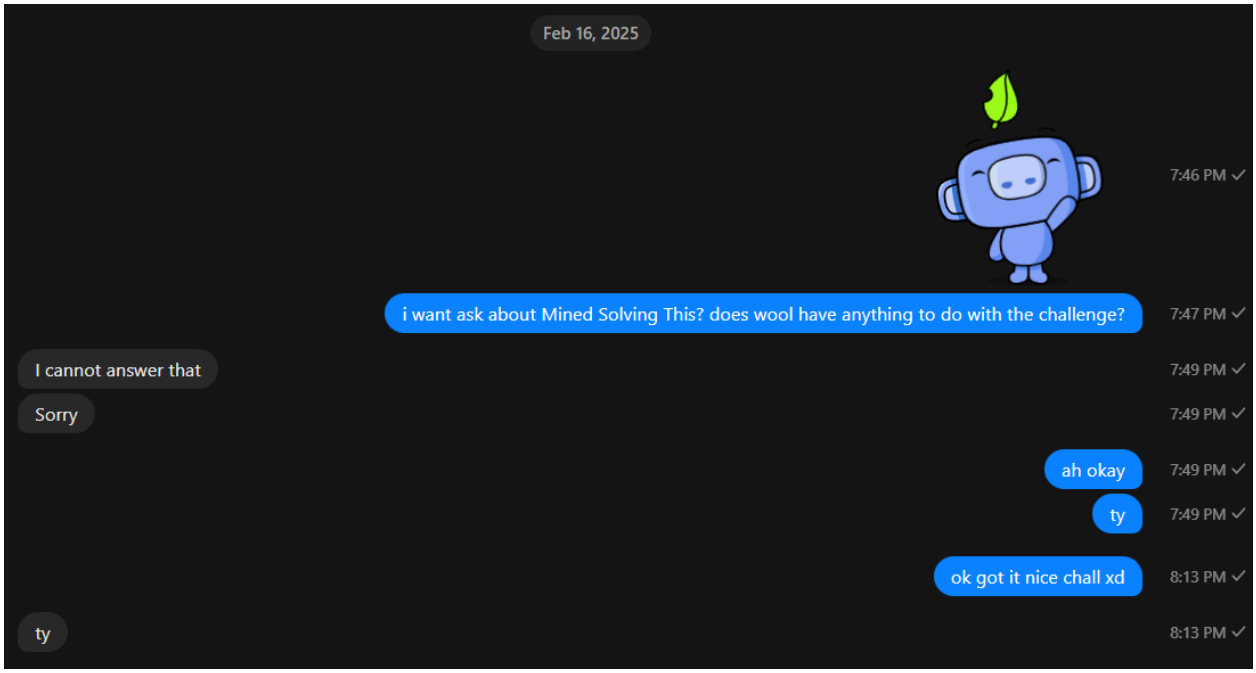
shwhale

I used to think Minecraft was just a silly game. I woold play it for hours and accomplish nothing. Making this challenge gave me perspective though!

Note: You do not need a copy of Minecraft for this challenge

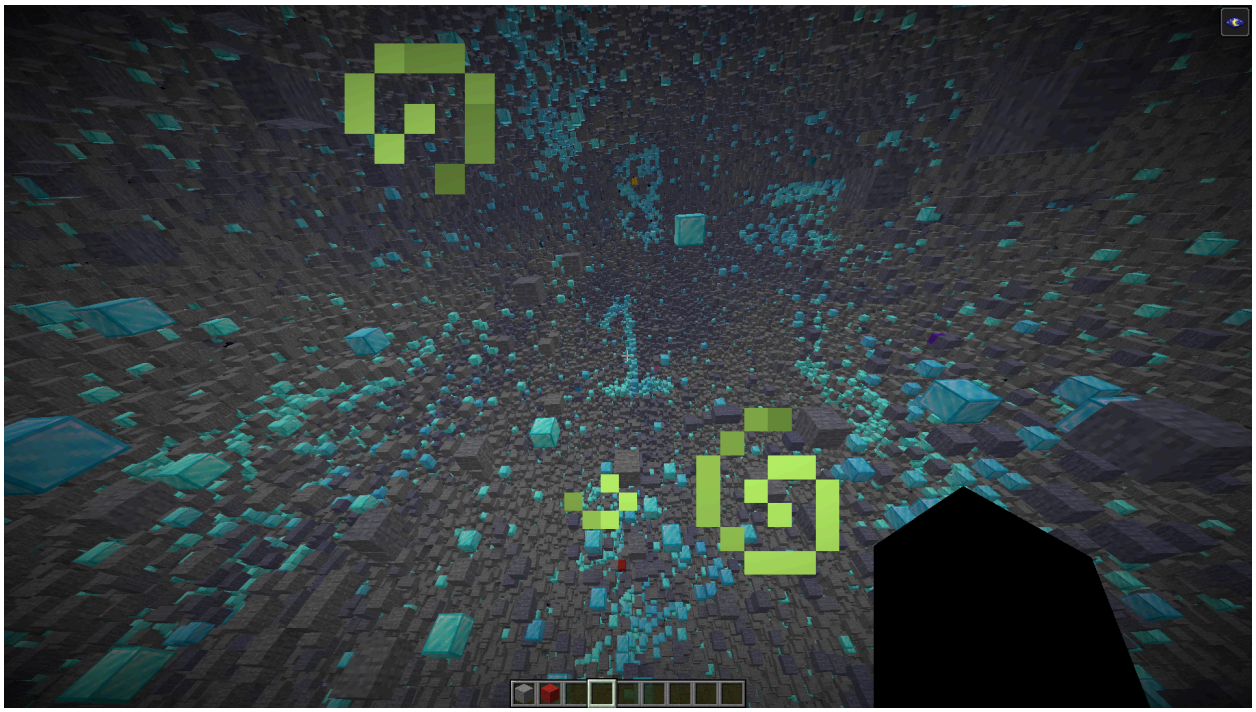
So this challenge is a **Minecraft** map. Downloaded the map and opened it in **Minecraft**. Open the map in **Minecraft** and we will be spawned in a far place.

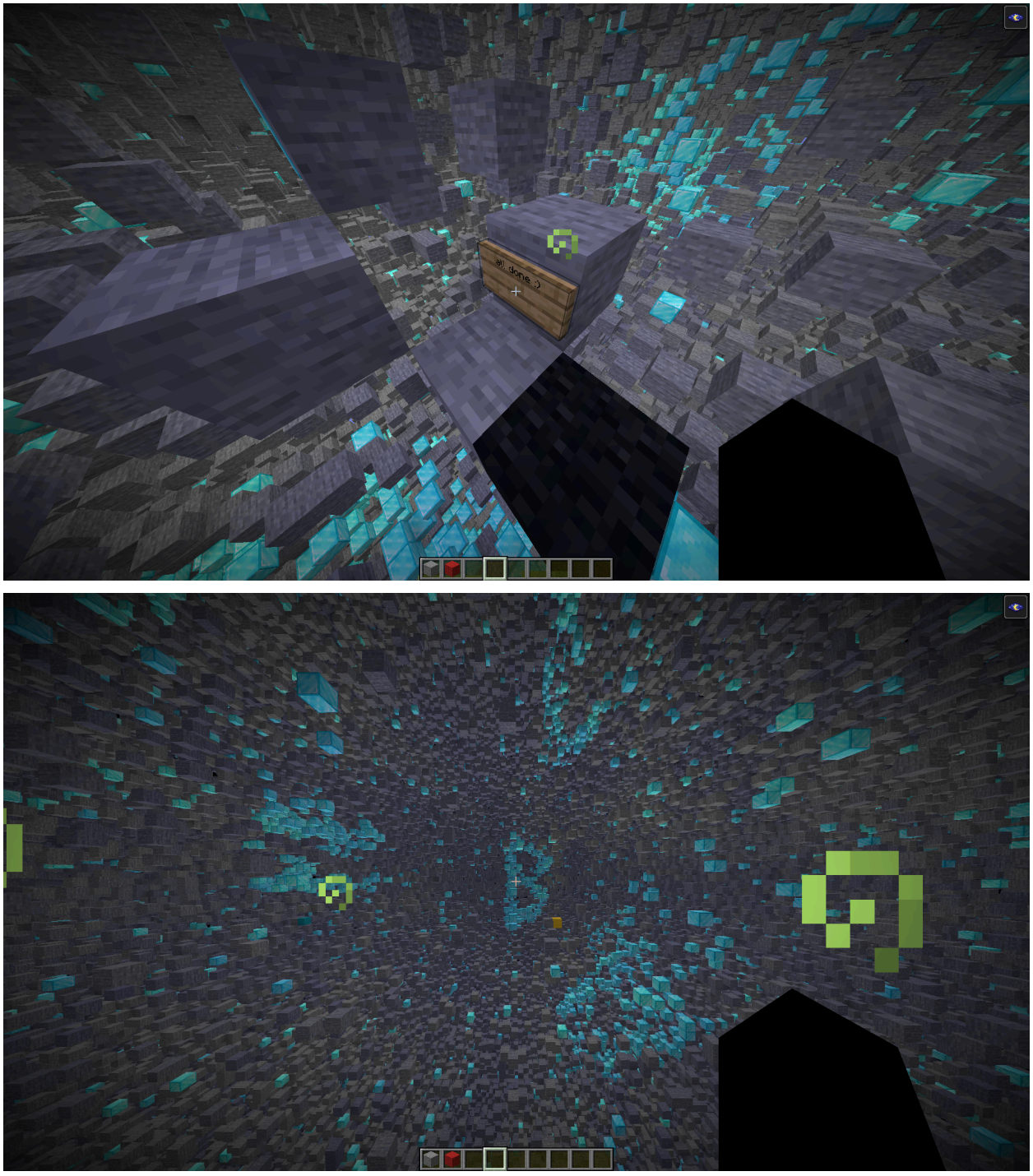
At the beginning we will be at coordinates `1000 ~ 1000`, so we can teleporte to coordinates `0 ~ 0` with the command `/tp 0 0 0`. After that looked around and found a structure. Because there were no more clues, I tried to open the inventory and see if there were any clues there. And it turned out there were clues there.



After asking the problem setter, and he couldn't answer it was certain that `wool` was a clue. And it turned out to be true after explored the structure, it turned out that there were several colored `wool` blocks after standing on the `wool` and looking around, something would happen! namely

the `diamond` block will show a letter pattern. And after we collect all the letter patterns, we will get a flag.





By matching the pattern with the `wool` order in the inventory, we will get the flag.

Flag: `bronco{m1n3d}`

OSINT

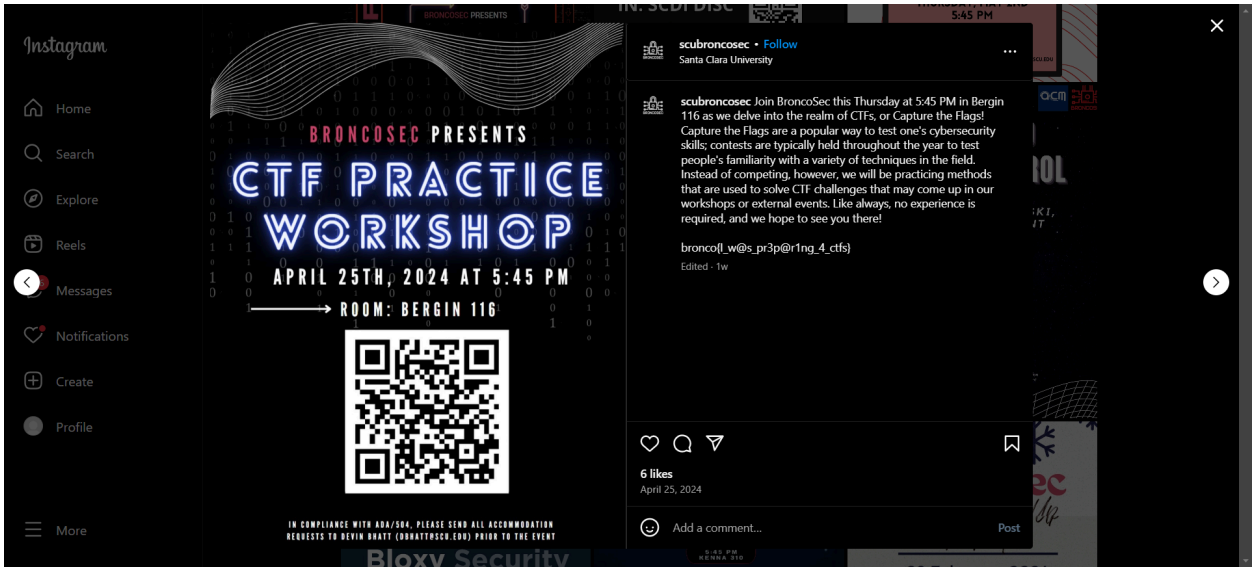
April 25

10

tiffany_ttn

Do you really want to know where BroncoSec (full of security enthusiasts and influencers) was April 25?

There is Instagram account `scubroncosec` that has a post on **April 25**.



Flag: `bronco{I_w@s_pr3p@r1ng_4_ctfs}`

Elite Stacker

266

yoshie878

YOOOO check out this new personal best in sprint that my bestie yoshie just got! He truly is the elitest elite stacker of all time! (p.s. not really. he needs *waaaay* more T-Spins.)

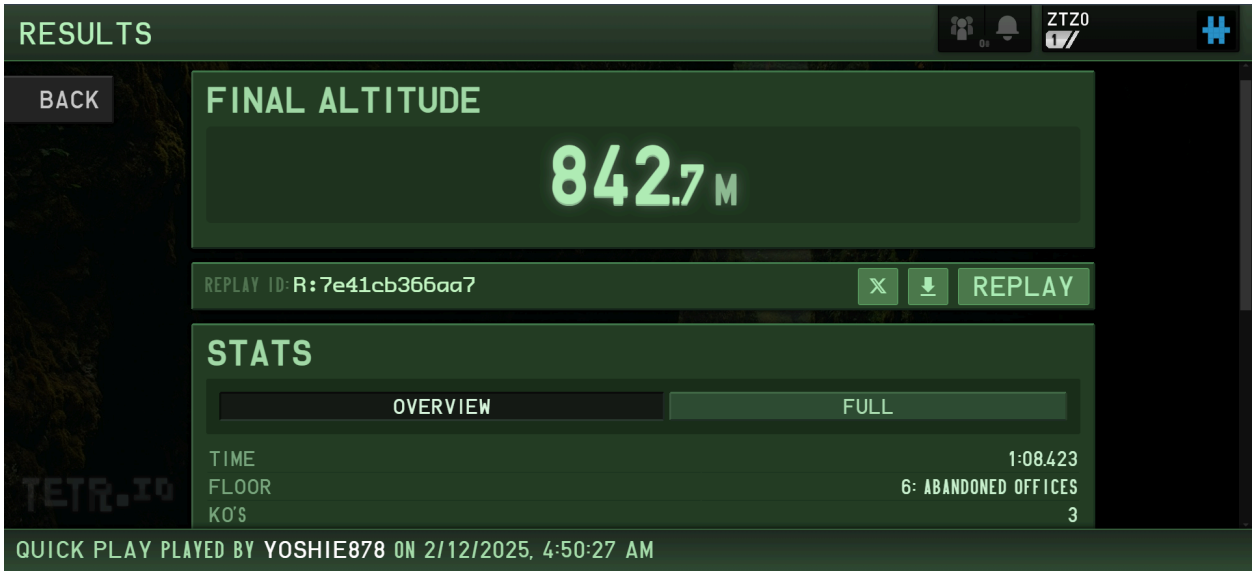
Ok honestly though, that mode is booooooring. Where's all the garbage at? No challenge at all.

Tell me, what is the highest floor and altitude in the Zenith Tower that yoshie has climbed up to?

Flag Format: `bronco{<Floor Name> <Altitude>}`, where:

- `Floor Name` is the name (NOT number) of the floor. Include space(s).
- `Altitude` is a number to one decimal place, commas optional, with `m` at the end for meters.
- There is a space between the entries.

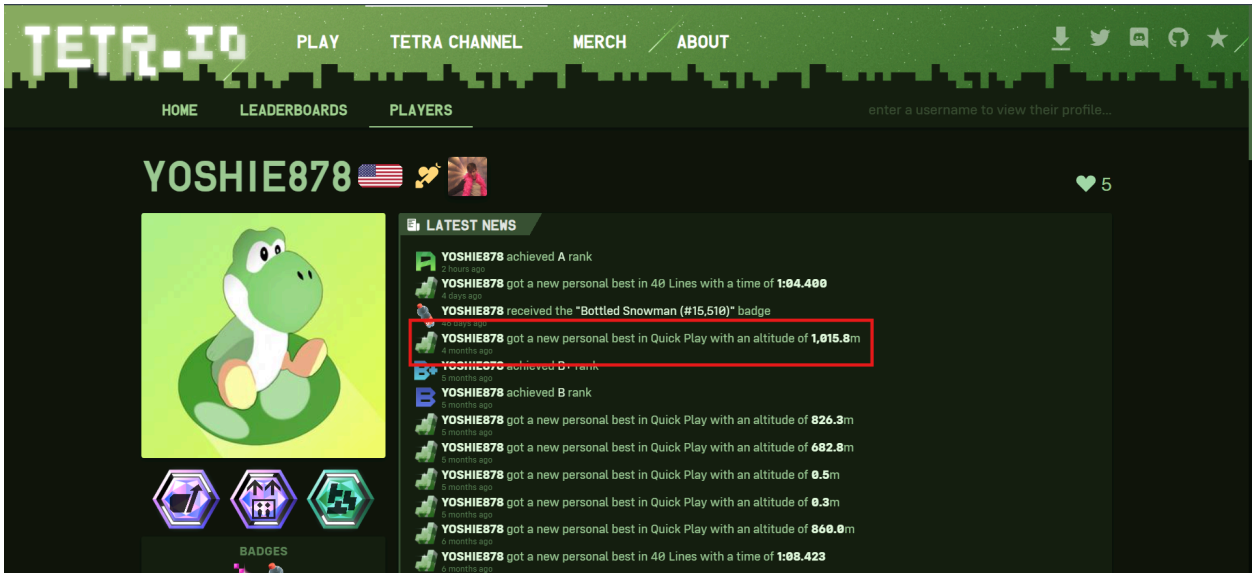
So they give a file named `imthegoatatstacking_7e41cb366aa7.ttr` which is a replay file for the game tetr.io. I downloaded the file and opened it in the game. The game is a tetris game and the replay file is a recording of a game played by the user. The user in this case is `yoshie878`.



The user mentions that he is the "elitest elite stacker of all time" and that he needs "waaaay more T-Spins". This is a hint that the user is good at the game and that he is not satisfied with his current performance.

The user also mentions that the mode is boring and that there is no challenge at all. This is a hint that the user is playing in a mode that is not challenging enough for him.

The user asks for the highest floor and altitude in the **Zenith Tower** that he has climbed up to. This is a hint that the user has played in the **Zenith Tower** mode and has reached a high floor and altitude.



By viewing the player's profile, we can see that the user has played in the **Zenith Tower** mode and has reached a high floor and altitude. The highest altitude that the user has reached is **1,015.8m**.

Floors

Zenith Tower, the stage of Quick Play, is divided into ten floors by altitude. Reaching a new floor for the first time, other than floors 1 and 10, unlocks a new mod.

Floor	Name	Height range(m)
1	Hall of Beginnings	0–50
2	The Hotel	0–150
3	The Casino	150–300
4	The Arena	300–450
5	The Museum	450–650
6	Abandoned Offices	650–850
7	The Laboratory	850–1100
8	The Core	1100–1350
9	Corruption	1350–1650
10	Platform of the Gods	1650 and above[QP note 2]

Because the replay file is expired, we cannot view the replay of the game. However, we can search the floor name in Google and find the altitude of the floor. Which is **The Laboratory** floor.

Flag: `bronco{The Laboratory 1,015.8m}`

Reverse Engineering

Reversing for Ophidiophiles

10

shwhale

Do you love python? Or at least tolerate it? Then this is the challenge for you!

When run with the correct flag, the given file prints:
`23a326c27bee9b40885df97007aa4dbe410e93`.

What is the flag?

The challenge provides a python script that encrypts the flag and prints the ciphertext. The script is as follows:

```
flag = input()
carry = 0
key = "Awesome!"
output = []
for i,c in enumerate(flag):
    val = ord(c)
    val += carry
    val %= 256
    val ^= ord(key[i % len(key)])
    output.append(val)
    carry += ord(c)
    carry %= 256

print(bytes(output).hex())
```

So here the flag is added to the carry where at the beginning the carry is 0 after that there will be a check if the flag is outside the ascii then the modulus is 256 after adding the carry then it will be continued with the XOR operation with the Awesome key! after that it will be added to the output array variable. After that the variable will be added with the ascii of the current character if the carry is outside the ascii range then it will be modulated by 256.

To decrypt the flag, we can reverse the encryption process. We can start by converting the ciphertext to bytes, and then decrypting each character in reverse order. The decryption process is as follows:

```
ciphertext = "23a326c27bee9b40885df97007aa4dbe410e93"
ciphertext = bytes.fromhex(ciphertext)
key = "Awesome!"
carry = 0
```

```
flag = []
for i, c in enumerate(ciphertext):
    c ^= ord(key[i % len(key)])
    c -= carry
    c %= 256
    flag.append(c)
    carry += c
    carry %= 256

print(f'Flag: {bytes(flag).decode()}')
```

Flag: bronco{charge_away}

theflagishere!

10

serilical

So, my friend sent me this program that's supposed to determine the flag for this challenge, right? But, somehow, they forgot to actually say what the flag is. Classic move. 😂 Now, it's on you to figure out what the true flag is. If you can crack it and figure out what my friend was trying to send, that flag is all yours! Ready to flex those decoding skills? Let's get it!

Format: bronco{flag}

Given a file named `theflagishere.pyc` and a hint that it is a python compiled file, we can use a decompiler to get the source code. I used pylingual.io to decompile the file.

https://pylingual.io/view_chimera?identifier=1f1468a7d8ca8cbfa686caa36e17a1a373bb9fb326870e9a78e2655dc7a8fce6

```
# Decompiled with PyLingual (https://pylingual.io)
# Internal filename: theflagishere.py
# Bytecode version: 3.9.0beta5 (3425)
# Source timestamp: 2025-02-13 23:36:18 UTC (1739489778)

def what_do_i_do(whoKnows):
    a_st = {}
    for a in whoKnows:
        if a_st.get(a) == None:
            a_st[a] = 1
        else:
            a_st[a] += 1
    variable_name = 0
    not_a_variable_name = 'None'
    for a in a_st:
        if a_st[a] > variable_name:
            not_a_variable_name = a
            variable_name = a_st[a]
    return (not_a_variable_name, variable_name)

def char_3():
    return 'm'

def i_definitely_return_the_flag():
```

```
    def notReal():

        def actually_real():
            return 'actuallyaflag'
        return actually_real

    def realFlag():
        return 'xXx___this__is_the__flag___xXx'
    return (realFlag, notReal)

def i_am_a_function_maybe(param):
    variableName = (param + 102) * 47
    for i in range(0, 100):
        variableName *= i + 1
        variableName /= i + 1
        newVariable = variableName * i
        newVariable += 100
    return chr(ord(chr(int(variableName) + 1)))

def i_do_not_know():
    realFlagHere = 'br0nc0s3c_fl4g5_4r3_345y'
    return 'long_live_long_flags'

def unrelated_statement():
    return 'eggs_go_great_with_eggs'

def i_am_a_function(param):
    variableName = (param + 102) * 47
    for i in range(0, 100):
        variableName *= i + 1
        newVariable = variableName * i
        newVariable += 100
        variableName /= i + 1
    return chr(ord(chr(int(variableName))))

def i_return_a_helpful_function():

    def i_do_something(char):
        var = []
        for i in range(54, 2000):
            var.append(ord(char) / 47 - 102)
        var.reverse()
        return var.pop()
    return i_do_something

def i_return_the_flag():
    return 'thisisdefinitelytheflag!'
```

```
def i():
    return 'free_flag_f'

def char_0():
    return
i_am_a_function_maybe(i_return_a_helpful_function()(what_do_i_do(i_return_the_flag())[
0]))

def char_1_4_6():
    return
i_am_a_function_maybe(i_return_a_helpful_function()(what_do_i_do(i_definitely_return_t
he_flag()[0]())[0]))

def char_2_5_9():
    return
i_am_a_function_maybe(i_return_a_helpful_function()(what_do_i_do(i_definitely_return_t
he_flag()[1]()())[0]))

def char_7():
    return
i_am_a_function_maybe(i_return_a_helpful_function()(what_do_i_do(interesting()()()())[
0]))

def char_8():
    return
i_am_a_function_maybe(i_return_a_helpful_function()(what_do_i_do(i_do_not_know())[0]))

def char_10():
    return
i_am_a_function_maybe(i_return_a_helpful_function()(what_do_i_do(unrelated_statement()
)[0]))

def interesting():

    def notinteresting():

        def veryuninteresting():

            def interesting_call():
                return i
            return interesting_call
        return veryuninteresting
    return notinteresting
```

There are a lot of functions that are not used in the main function. We can ignore them. The main function is `char_0`, `char_1_4_6`, `char_2_5_9`, `char_7`, `char_8`, and `char_10`. We can run these functions to get the flag.

```
print(f'bronco{{{char_0() + char_1_4_6() + char_2_5_9() + char_3() + char_1_4_6() + char_2_5_9() + char_6() + char_1_4_6() + char_2_5_9() + char_9() + "?" + char_8() + char_10()}}}')
```

Because `char_7` had an error, `***<module>.char_7: Failure: Different bytecode` we need to guess the character for `char_7` which is `f`.

Flag: `bronco{i_am_a_flag}`

sus

255

serilical

So, my friend just hit me up with this lit file, right? But like, they totally forgot to define their keywords—straight up left me hanging. It's like I'm playing a game of "guess the meaning" with no map. I gotta reverse the code and figure out what's actually poppin' in there. There's a flag hidden somewhere, but it's got mad twists and turns. If you're down to decode it too, try flipping the meaning of the words and see if you can catch the vibe. Let's see who can crack it first!

The challenge provides a `sus.cpp` file with the following contents we need to figure out the meaning of the keywords:

```
#define skibidi ?

using namespace skibidi;

#define hawk ?
#define pressed ?
#define crash_out ?
#define ate ?
#define twin ?
#define periodt ?
#define vibe ?
#define blud ?
#define delulu ?
#define uhh ?
#define slay ?
#define dap ?
#define yap ?
#define diff ?
#define lit ?
#define free ?
#define stan ?
#define savage ?
#define hop_off ?
#define take_a_seat ?
#define amped ?
#define tuah ?
#define gucci ?
#define finna ?
#define rent ?
```

```
#define tea ?
#define flex ?
#define mid ?
#define cancelled ?
```

AFTER SOME MANUAL WORK, we can figure out the following mapping:

```
#define skibidi std

using namespace skibidi;

#define hawk if
#define pressed -
#define crash_out }
#define ate +
#define twin ==
#define periodt ;
#define vibe while
#define blud char
#define delulu %
#define uhh ,
#define slay return
#define dap {
#define yap cout
#define diff <
#define lit int
#define free ]
#define stan =
#define savage NULL
#define hop_off )
#define take_a_seat endl
#define amped *
#define tuah else
#define gucci string
#define finna (
#define rent [
#define tea <<
#define flex >
#define mid /
#define cancelled break
```

Flag: `bronco{br4inr0t}`

Actual Reversing

430

shwhale

Here is something reversible. :)

ok!! **Actual Reversing**.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char i; // [rsp+7h] [rbp-59h]
  int v5; // [rsp+8h] [rbp-58h]
  int v6; // [rsp+Ch] [rbp-54h]
  char s[72]; // [rsp+10h] [rbp-50h] BYREF
  unsigned __int64 v8; // [rsp+58h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  puts("Welcome to the transformer! We take what you have, and make it into what you
have always wanted!");
  printf("What do you have to offer?\n> ");
  fgets(s, 64, _bss_start);
  s[strcspn(s, "\n")] = 0;
  v5 = 0;
  v6 = 0;
  while ( s[v5] )
  {
    for ( i = s[v5]; i; i >>= 1 )
      v6 += i & 1;
    ++v5;
  }
  if ( v6 == 108 )
  {
    puts("Here's my perscription:");
    perscribe(s);
    printf("%s", TRUTH);
  }
  else
  {
    puts("That'll NEVER turn into what you want!");
  }
  return 0;
}
```

The binary reads a string from the user, and then calculates the number of bits set in the string. If the number of bits set is 108, it calls a function `perscribe` with the string. The `perscribe` function then does some bit manipulation and prints out a message.

```
unsigned __int64 __fastcall perscribe(char *a1)
{
  void *v1; // rsp
  __int64 v3; // [rsp+0h] [rbp-60h] BYREF
  char *s; // [rsp+8h] [rbp-58h]
  int i; // [rsp+18h] [rbp-48h]
  int j; // [rsp+1Ch] [rbp-44h]
  int k; // [rsp+20h] [rbp-40h]
  int v8; // [rsp+24h] [rbp-3Ch]
  int v9; // [rsp+28h] [rbp-38h]
  int v10; // [rsp+2Ch] [rbp-34h]
  int v11; // [rsp+30h] [rbp-30h]
  int v12; // [rsp+34h] [rbp-2Ch]
  __int64 v13; // [rsp+38h] [rbp-28h]
  __int64 *v14; // [rsp+40h] [rbp-20h]
  unsigned __int64 v15; // [rsp+48h] [rbp-18h]

  s = a1;
  v15 = __readfsqword(0x28u);
  v8 = strlen(TRUTH);
  v9 = 8 * v8;
  v13 = 8 * v8 - 1LL;
  v1 = alloca(16 * ((8 * v8 + 15LL) / 0x10uLL));
  v14 = &v3;
  for ( i = 0; i < 8 * v8; ++i )
    *((_BYTE *)v14 + i) = 0;
  v10 = strlen(s);
  for ( i = 0; i < v10; ++i )
  {
    for ( j = 0; j <= 7; ++j )
    {
      v11 = 1 << j;
      if ( ((1 << j) & s[i]) != 0 )
      {
        v12 = 8 * i + j;
        for ( k = 0; ((TRUTH[k >> 3] >> (k & 7)) & 1) == 0 || *((_BYTE *)v14 + k); ++k
)
          ;
        printf("Take %d of these, then\n", (unsigned int)(k - v12));
        *((_BYTE *)v14 + k) = 1;
      }
    }
  }
```

```
  puts("You're done!");
  return v15 - __readfsqword(0x28u);
}
```

The `perscribe` function initializes an array of bytes `v14` with zeros. It then iterates over each byte in the input string `s`, and for each bit set in the byte, it finds the first bit set in the `TRUTH` string that has not already been used. It then prints out `"Take X of these, then"`, where `X` is the difference between the bit position in the `TRUTH` string and the bit position in the input string. It then marks that bit as used.

```
.data:0000000000004040 TRUTH           dq offset aCanBirdsEvenUn
.data:0000000000004040                                         ; DATA XREF: main+E8↑r
.data:0000000000004040                                         ; perscribe+22↑r ...
.data:0000000000004040 _data           ends                    ; "Can birds even understand me?"
```

Matching with `TRUTH`

- The function scans through the bits of `TRUTH` (`for (k = 0; ... ; ++k)`) looking for:
  - The first bit that is set (`((TRUTH[k >> 3] >> (k & 7)) & 1) != 0`).
  - The first bit that has not already been used (`!*((_BYTE *)v14 + k)`).
- Once it finds such a bit, it prints `"Take X of these, then"`, where `X = k - v12` (the offset between input and `TRUTH`).
- It marks that bit position as used (`*((_BYTE *)v14 + k) = 1`).

They give `perscription` for the correct input string. We can use this `perscription` to recover the flag.

```
Here's my perscription:
Take -1 of these, then
Take -4 of these, then
Take 0 of these, then
Take -1 of these, then
Take 1 of these, then
Take 1 of these, then
Take 3 of these, then
Take 2 of these, then
Take 2 of these, then
Take 3 of these, then
Take 3 of these, then
Take 8 of these, then
Take 11 of these, then
Take 12 of these, then
Take 12 of these, then
Take 13 of these, then
```

```
Take 14 of these, then
Take 15 of these, then
Take 14 of these, then
Take 16 of these, then
Take 15 of these, then
Take 15 of these, then
Take 14 of these, then
Take 17 of these, then
Take 19 of these, then
Take 19 of these, then
Take 19 of these, then
Take 19 of these, then
Take 20 of these, then
Take 20 of these, then
Take 19 of these, then
Take 25 of these, then
Take 27 of these, then
Take 28 of these, then
Take 28 of these, then
Take 26 of these, then
Take 28 of these, then
Take 28 of these, then
Take 28 of these, then
Take 28 of these, then
Take 26 of these, then
Take 27 of these, then
Take 25 of these, then
Take 27 of these, then
Take 26 of these, then
Take 28 of these, then
Take 28 of these, then
Take 27 of these, then
Take 28 of these, then
Take 26 of these, then
Take 32 of these, then
Take 31 of these, then
Take 30 of these, then
Take 31 of these, then
Take 31 of these, then
Take 30 of these, then
Take 31 of these, then
Take 30 of these, then
Take 30 of these, then
Take 29 of these, then
Take 28 of these, then
Take 29 of these, then
Take 31 of these, then
```

```
Take 28 of these, then
Take 27 of these, then
Take 28 of these, then
Take 29 of these, then
Take 29 of these, then
Take 31 of these, then
Take 33 of these, then
Take 33 of these, then
Take 32 of these, then
Take 32 of these, then
Take 32 of these, then
Take 32 of these, then
Take 32 of these, then
Take 29 of these, then
Take 32 of these, then
Take 33 of these, then
Take 32 of these, then
Take 32 of these, then
Take 28 of these, then
Take 32 of these, then
Take 30 of these, then
Take 31 of these, then
Take 30 of these, then
Take 30 of these, then
Take 31 of these, then
Take 30 of these, then
Take 33 of these, then
Take 35 of these, then
Take 33 of these, then
Take 39 of these, then
Take 37 of these, then
Take 37 of these, then
Take 37 of these, then
Take 37 of these, then
Take 37 of these, then
Take 38 of these, then
Take 39 of these, then
Take 41 of these, then
Take 41 of these, then
Take 40 of these, then
Take 39 of these, then
Take 39 of these, then
Take 39 of these, then
Take 39 of these, then
Take 39 of these, then
You're done!
Can birds even understand me?
```

The `perscription` tells us how to construct the flag from the input string. We can write a script to do this.

```python
import math

TRUTH = "Can birds even understand me?"

T = []
for k in range(8 * len(TRUTH)):
    if (ord(TRUTH[k // 8]) >> (k % 8)) & 1:
        T.append(k)

log_d = [
    -1, -4, 0, -1, 1, 1, 3, 2, 2, 3, 3, 8, 11, 12, 12, 13, 14, 15, 14, 16,
    15, 15, 14, 17, 19, 19, 19, 19, 20, 20, 19, 25, 27, 28, 28, 26, 28, 28,
    28, 28, 26, 27, 25, 27, 26, 28, 28, 27, 28, 26, 32, 31, 30, 31, 31, 30,
    31, 30, 30, 29, 28, 29, 31, 28, 27, 28, 29, 29, 31, 33, 33, 32, 32, 32,
    32, 32, 29, 32, 33, 32, 32, 28, 32, 30, 31, 30, 30, 31, 30, 33, 35, 33,
    39, 37, 37, 37, 37, 37, 38, 39, 41, 41, 40, 39, 39, 39, 39, 39
]

b_positions = []
for j, d in enumerate(log_d):
    b_j = T[j] - d
    b_positions.append(b_j)

max_bit = max(b_positions)
num_bytes = math.ceil((max_bit + 1) / 8)

input_bytes = [0] * num_bytes

for pos in b_positions:
    byte_index = pos // 8
    bit_index = pos % 8
    input_bytes[byte_index] |= (1 << bit_index)

print(f'Flag: {"".join(chr(b) for b in input_bytes)}')
```

Flag: `bronco{r3v3r5ed_3n0ugh?}`