Girls in CTF 2024

Writeup by BL4CKP!NKH4T

pinx
sushi

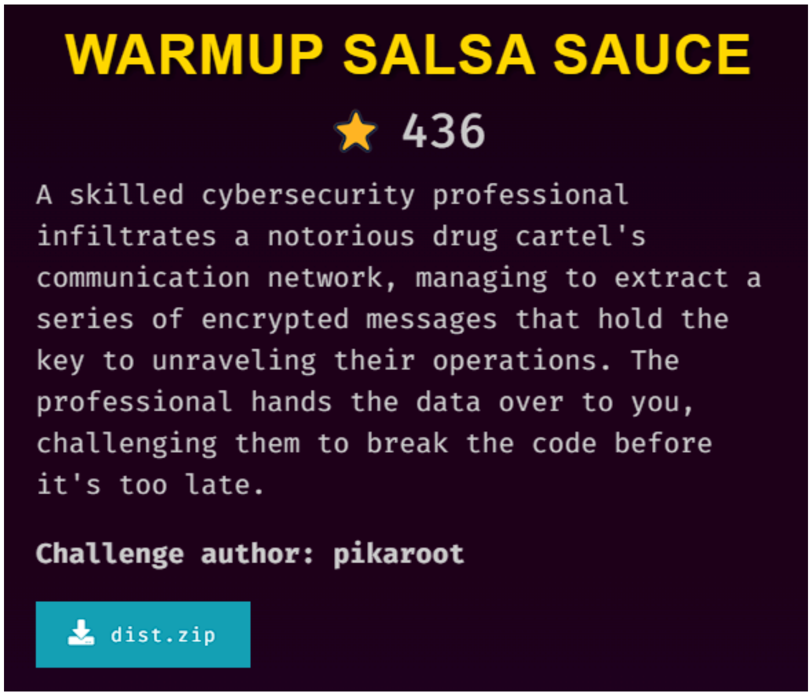
12 October 2024

Girls in CTF 2024 was… a humbling experience! I honestly thought it would be a breeze, just like last year’s GCTF, but I was so wrong. I got HUMBLED, no doubt about it. The categories and challenges really pushed us (and our machine specs) to dig deeper and put in that extra effort. Initially, we aimed for a spot in the top 3, but after facing the challenges head-on, we were like, “GG!” Hahaha, our new goal? Just soak up new knowledge (and trust me, we did!!). This was our first time attempting blockchain questions phew. But hey, I’m super grateful we landed in 11th place! Unfortunately there were a lot of questions we attempted but were unsuccessful (some 95% close to getting the flag but we got stuck there till the end T-T). Hence the challenges below were the only flags that we managed to capture. Enjoy the short read!

CRYPTOGRAPHY

WARMUP SALSA SAUCE

Flag: `GCTF{y0u_f0und_th3_c0cain3_a7f9f6bdeabd34dde0fa3037284864eb}`



WARMUP SALSA SAUCE

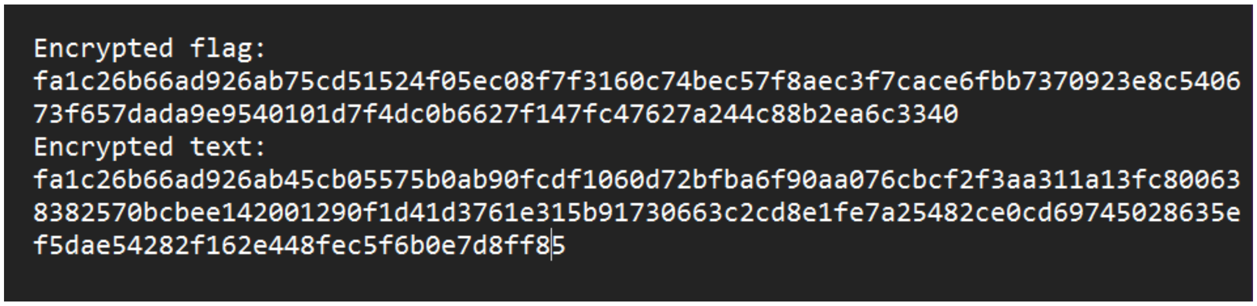
436

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

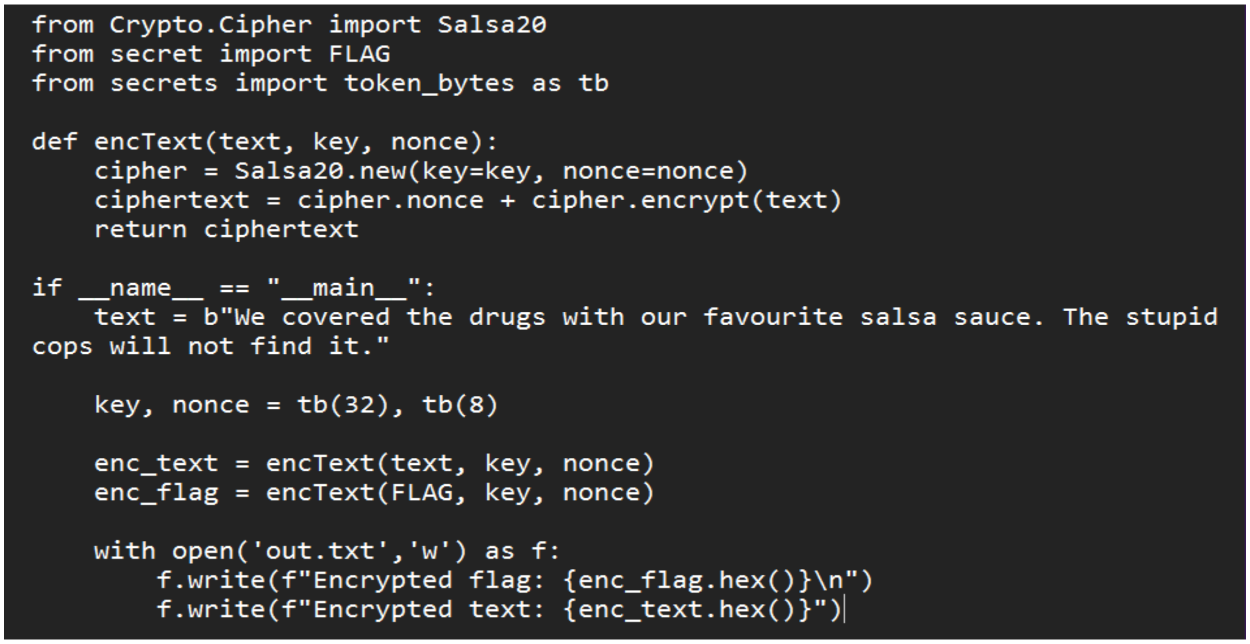
In this challenge, we have an encrypted flag and text in out.txt, using the Salsa20 method in the wss.py script.

out.txt:



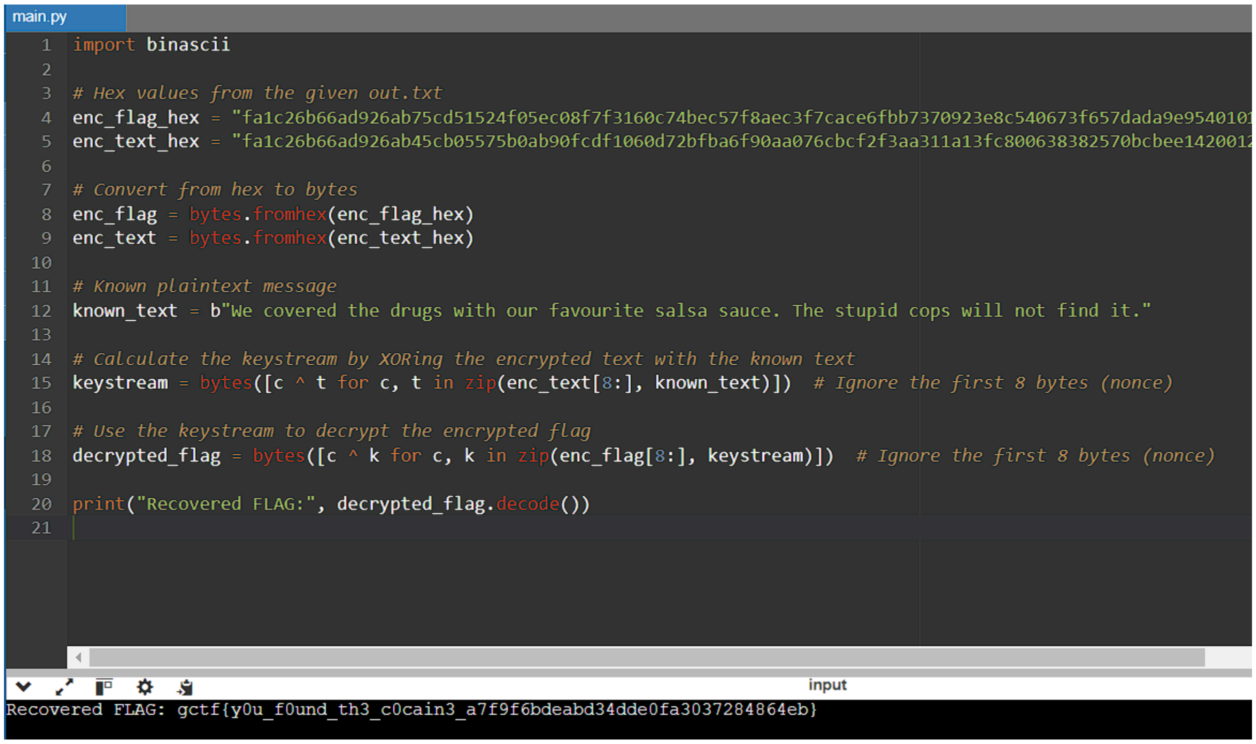
```
Encrypted flag:
fa1c26b66ad926ab75cd51524f05ec08f7f3160c74bec57f8aec3f7cace6fbb7370923e8c5406
73f657dada9e9540101d7f4dc0b6627f147fc47627a244c88b2ea6c3340
Encrypted text:
fa1c26b66ad926ab45cb05575b0ab90fcdf1060d72bfba6f90aa076cbcf2f3aa311a13fc80063
8382570bcbee142001290f1d41d3761e315b91730663c2cd8e1fe7a25482ce0cd69745028635e
f5dae54282f162e448fec5f6b0e7d8ff85
```

wss.py:



```python
from Crypto.Cipher import Salsa20
from secret import FLAG
from secrets import token_bytes as tb

def encText(text, key, nonce):
    cipher = Salsa20.new(key=key, nonce=nonce)
    ciphertext = cipher.nonce + cipher.encrypt(text)
    return ciphertext

if __name__ == "__main__":
    text = b"We covered the drugs with our favourite salsa sauce. The stupid
cops will not find it."

    key, nonce = tb(32), tb(8)

    enc_text = encText(text, key, nonce)
    enc_flag = encText(FLAG, key, nonce)

    with open('out.txt','w') as f:
        f.write(f"Encrypted flag: {enc_flag.hex()}\n")
        f.write(f"Encrypted text: {enc_text.hex()}")
```

The same key and nonce are used for both encryptions, generated by token_bytes(32) and token_bytes(8). We know the original text ("We covered the drugs with our favourite salsa sauce. The stupid cops will not find it.") and have its encrypted version, so we can try to reverse the encryption to find the key and nonce. Once we have the key and nonce, we can use them to write a code to decrypt the flag.



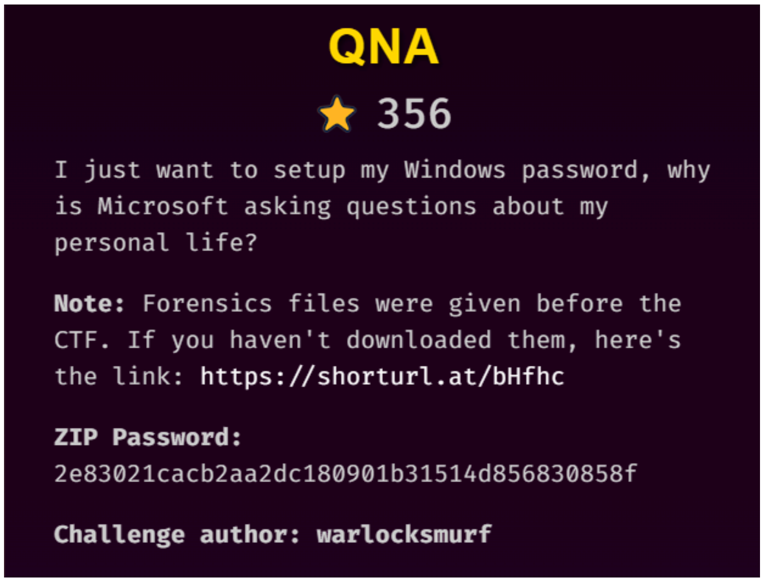
```python
import binascii

# Hex values from the given out.txt
enc_flag_hex = "fa1c26b66ad926ab75cd51524f05ec08f7f3160c74bec57f8aec3f7cace6fbb7370923e8c540673f657dada9e9540101
enc_text_hex = "fa1c26b66ad926ab45cb05575b0ab90fcdf1060d72bfba6f90aa076cbcf2f3aa311a13fc800638382570bcbee1420012

# Convert from hex to bytes
enc_flag = bytes.fromhex(enc_flag_hex)
enc_text = bytes.fromhex(enc_text_hex)

# Known plaintext message
known_text = b"We covered the drugs with our favourite salsa sauce. The stupid cops will not find it."

# Calculate the keystream by XORing the encrypted text with the known text
keystream = bytes([c ^ t for c, t in zip(enc_text[8:], known_text)])  # Ignore the first 8 bytes (nonce)

# Use the keystream to decrypt the encrypted flag
decrypted_flag = bytes([c ^ k for c, k in zip(enc_flag[8:], keystream)])  # Ignore the first 8 bytes (nonce)

print("Recovered FLAG:", decrypted_flag.decode())

```

```
Recovered FLAG: gctf{y0u_f0und_th3_c0cain3_a7f9f6bdeabd34dde0fa3037284864eb}
```

FORENSICS

QNA

Flag: `GCTF{p3rs0nal_s3cr3ts_r3g1stry}`


QNA

356

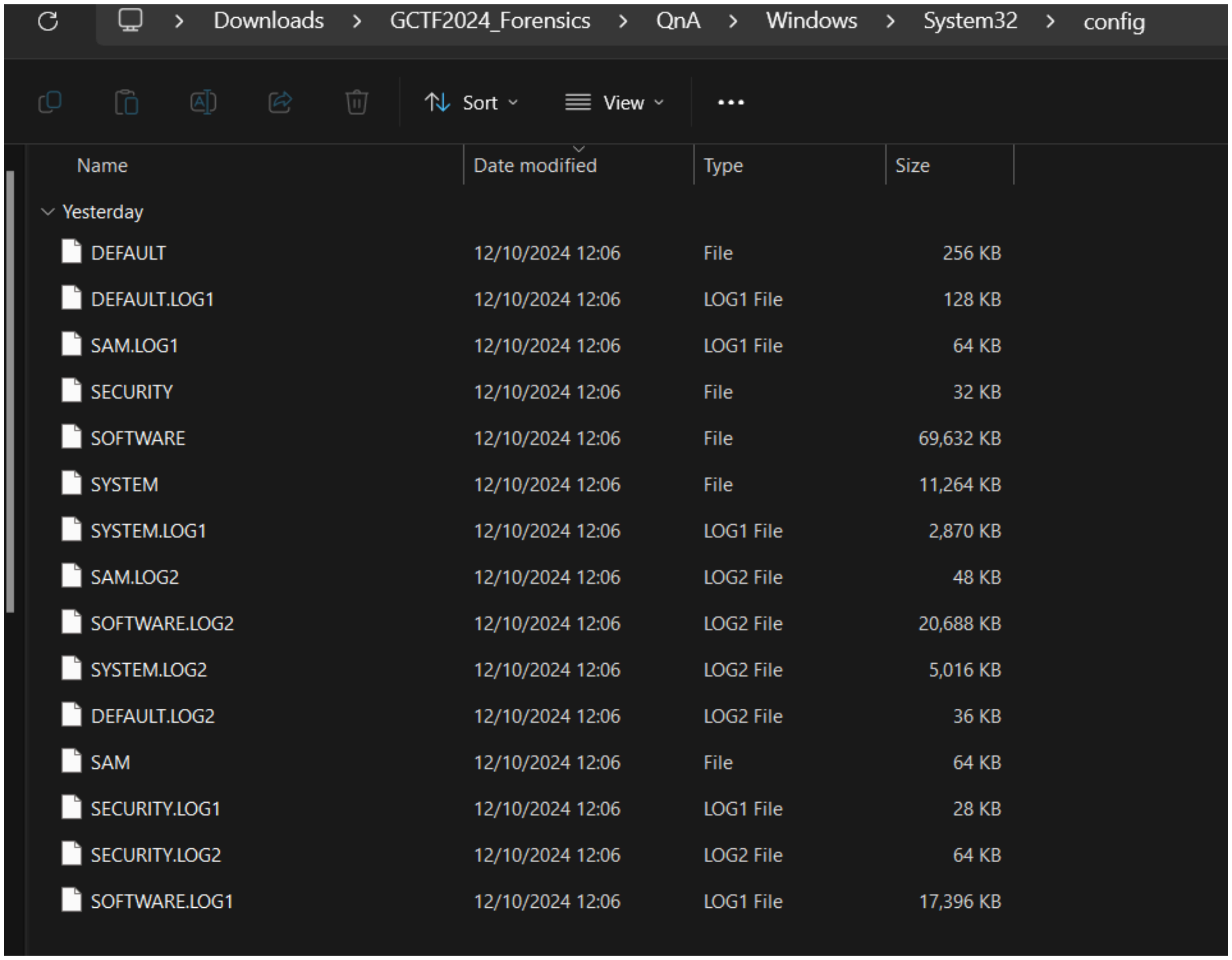
I just want to setup my Windows password, why is Microsoft asking questions about my personal life?

Note: Forensics files were given before the CTF. If you haven't downloaded them, here's the link: https://shorturl.at/bHfhc

ZIP Password: 2e83021cacb2aa2dc180901b31514d856830858f

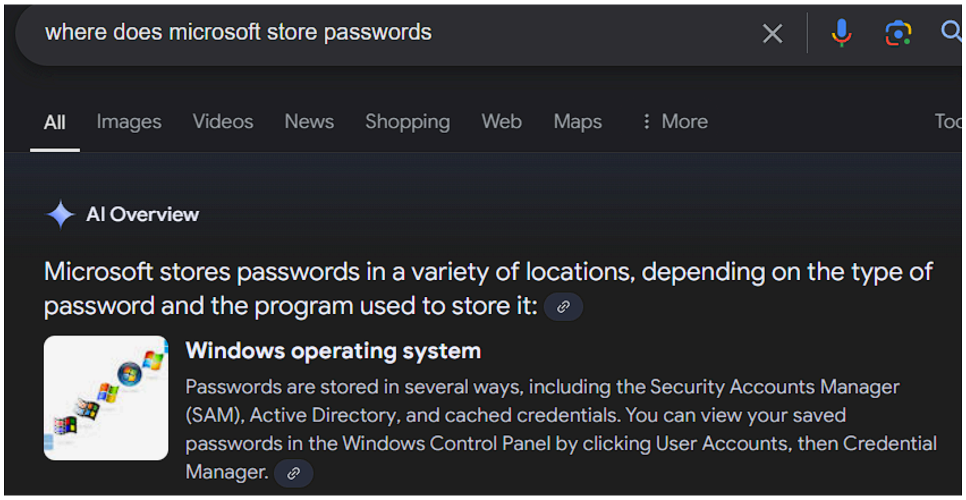
Challenge author: warlocksmurf

For this challenge, we were given a folder full of files like this that seem to contain personal information related to Microsoft account setups:


Downloads > GCTF2024_Forensics > QnA > Windows > System32 > config
Sort
View

Name	Date modified	Type	Size
Yesterday			
DEFAULT	12/10/2024 12:06	File	256 KB
DEFAULT.LOG1	12/10/2024 12:06	LOG1 File	128 KB
SAM.LOG1	12/10/2024 12:06	LOG1 File	64 KB
SECURITY	12/10/2024 12:06	File	32 KB
SOFTWARE	12/10/2024 12:06	File	69,632 KB
SYSTEM	12/10/2024 12:06	File	11,264 KB
SYSTEM.LOG1	12/10/2024 12:06	LOG1 File	2,870 KB
SAM.LOG2	12/10/2024 12:06	LOG2 File	48 KB
SOFTWARE.LOG2	12/10/2024 12:06	LOG2 File	20,688 KB
SYSTEM.LOG2	12/10/2024 12:06	LOG2 File	5,016 KB
DEFAULT.LOG2	12/10/2024 12:06	LOG2 File	36 KB
SAM	12/10/2024 12:06	File	64 KB
SECURITY.LOG1	12/10/2024 12:06	LOG1 File	28 KB
SECURITY.LOG2	12/10/2024 12:06	LOG2 File	64 KB
SOFTWARE.LOG1	12/10/2024 12:06	LOG1 File	17,396 KB

Dang, there are so many files! So, which one should we use? The question mentions a Windows password by Microsoft, but honestly, even we don't know. So, we asked Mr. Google:

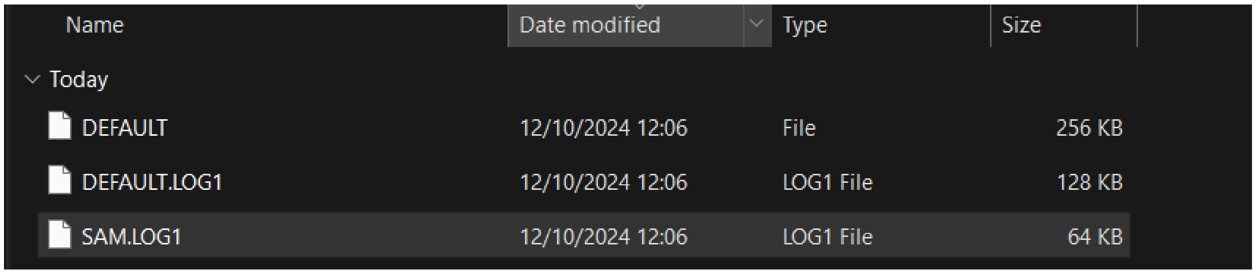


where does microsoft store passwords

Microsoft stores passwords in a variety of locations, depending on the type of password and the program used to store it:

Windows operating system
Passwords are stored in several ways, including the Security Accounts Manager (SAM), Active Directory, and cached credentials. You can view your saved passwords in the Windows Control Panel by clicking User Accounts, then Credential Manager.

Alright, Mr. Google said that the SAM file is where Microsoft stores the passwords. Since there are only three SAM files, it's manageable to check them one by one. After some searching, I found the flag in SAM.LOG1.



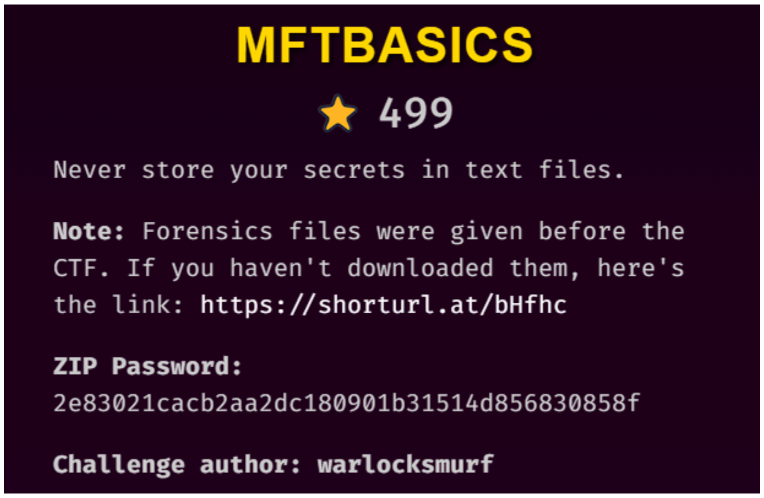
Name	Date modified	Type	Size
Today			
DEFAULT	12/10/2024 12:06	File	256 KB
DEFAULT.LOG1	12/10/2024 12:06	LOG1 File	128 KB
SAM.LOG1	12/10/2024 12:06	LOG1 File	64 KB

Btw, we used this online website to view our SAM file: https://filext.com/file-extension/SAM
Combine all the parts, and voila, you've got your flag! No offense, nothing personal ;)

```
 AdministratorsAdministrators have complete and unrestricted access to the computer/domain
 {"version":1,"questions":[{"question":"What was your first pet  s name?","answer":"GCTF{p3rs0nal"},
{"question":"What
 s the name of the city where you were born?","answer":"_s3cr3ts_"},{"question":"What was your childhood
nickname?","answer":"r3g1stry}"}]}
 Users Users are prevented from making accidental or intentional system-wide changes and can run most
applications
```

MFTBASICS

Flag: `GCTF{b4s1cs_0f_MFT}`



MFTBASICS

⭐ 499

Never store your secrets in text files.

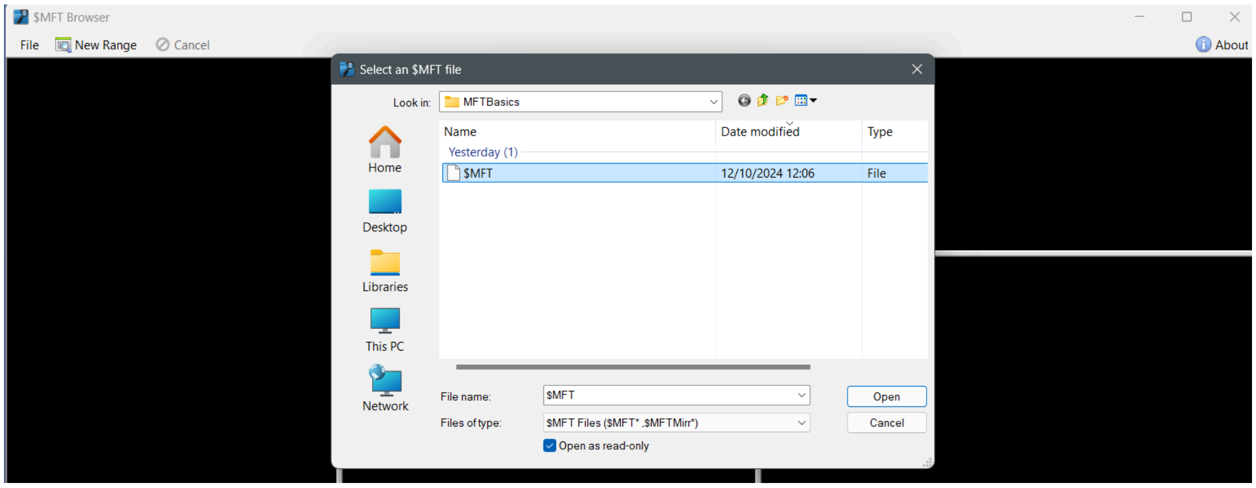
Note: Forensics files were given before the CTF. If you haven't downloaded them, here's the link: **https://shorturl.at/bHfhc**

ZIP Password:
2e83021cacb2aa2dc180901b31514d856830858f

Challenge author: warlocksmurf

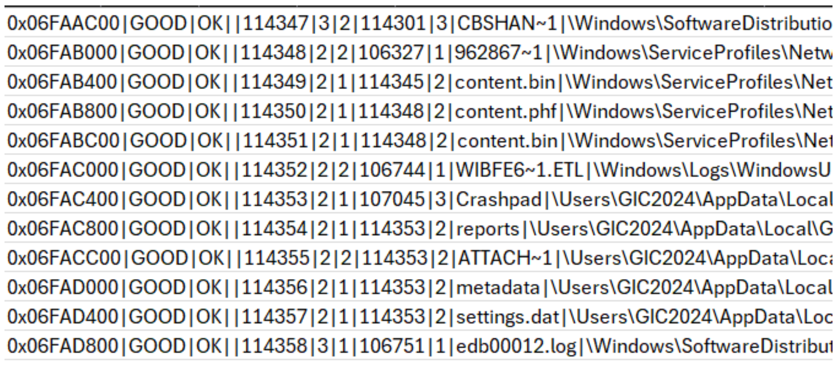
As the name suggests, this challenge is about the Master File Table (MFT). To prepare, I looked into MFT before tackling it. Many CTF challenges from other competitions recommended using MFT Browser, so I downloaded it from here:
https://github.com/kacos2000/MFT_Browser/releases/tag/v.1.0.80

After that, I loaded the $MFT file into the MFT Browser.

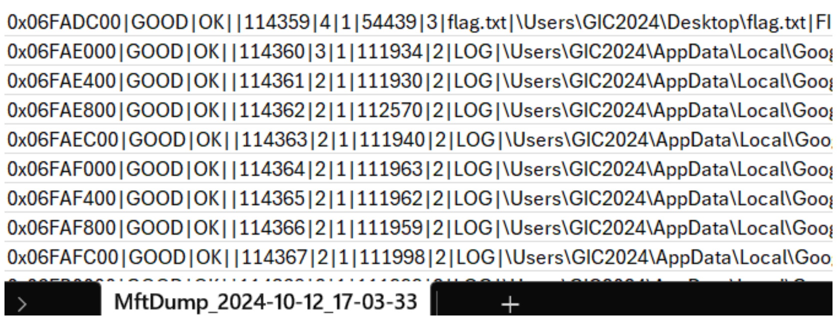


Since I wasn't sure how to search for flag.txt directly in the MFT Browser, I decided to convert the MFT file into a CSV format first. Here is the website to download the program: https://code.google.com/archive/p/mft2csv/downloads. Make sure to choose the "MFT to CSV (MFT2CSV)" option.

After converting, I searched for flag.txt to find its path.

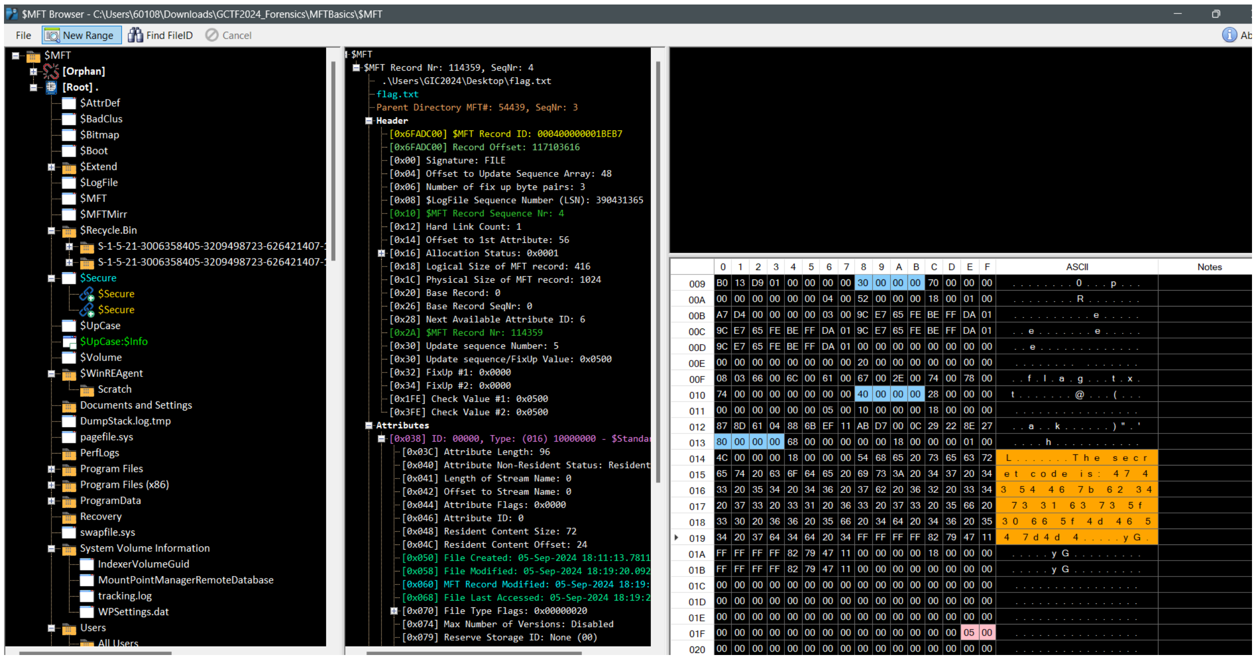


```
0x06FAAC00|GOOD|OK||114347|3|2|114301|3|CBSHAN~1|\Windows\SoftwareDistributio
0x06FAB000|GOOD|OK||114348|2|2|106327|1|962867~1|\Windows\ServiceProfiles\Netw
0x06FAB400|GOOD|OK||114349|2|1|114345|2|content.bin|\Windows\ServiceProfiles\Net
0x06FAB800|GOOD|OK||114350|2|1|114348|2|content.phf|\Windows\ServiceProfiles\Net
0x06FABC00|GOOD|OK||114351|2|1|114348|2|content.bin|\Windows\ServiceProfiles\Net
0x06FAC000|GOOD|OK||114352|2|2|106744|1|WIBFE6~1.ETL|\Windows\Logs\WindowsU
0x06FAC400|GOOD|OK||114353|2|1|107045|3|Crashpad|\Users\GIC2024\AppData\Local
0x06FAC800|GOOD|OK||114354|2|1|114353|2|reports|\Users\GIC2024\AppData\Local\G
0x06FACC00|GOOD|OK||114355|2|2|114353|2|ATTACH~1|\Users\GIC2024\AppData\Loc
0x06FAD000|GOOD|OK||114356|2|1|114353|2|metadata|\Users\GIC2024\AppData\Local
0x06FAD400|GOOD|OK||114357|2|1|114353|2|settings.dat|\Users\GIC2024\AppData\Loc
0x06FAD800|GOOD|OK||114358|3|1|106751|1|edb00012.log|\Windows\SoftwareDistribut
```



```
0x06FADC00|GOOD|OK||114359|4|1|54439|3|flag.txt|\Users\GIC2024\Desktop\flag.txt|Fl
0x06FAE000|GOOD|OK||114360|3|1|111934|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAE400|GOOD|OK||114361|2|1|111930|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAE800|GOOD|OK||114362|2|1|112570|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAEC00|GOOD|OK||114363|2|1|111940|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAF000|GOOD|OK||114364|2|1|111963|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAF400|GOOD|OK||114365|2|1|111962|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAF800|GOOD|OK||114366|2|1|111959|2|LOG|\Users\GIC2024\AppData\Local\Goo
0x06FAFC00|GOOD|OK||114367|2|1|111998|2|LOG|\Users\GIC2024\AppData\Local\Goo
```

Next, I reopened the MFT Browser and followed the path to flag.txt that I found in the CSV file. There, I found the hex value which tells me the secret code.



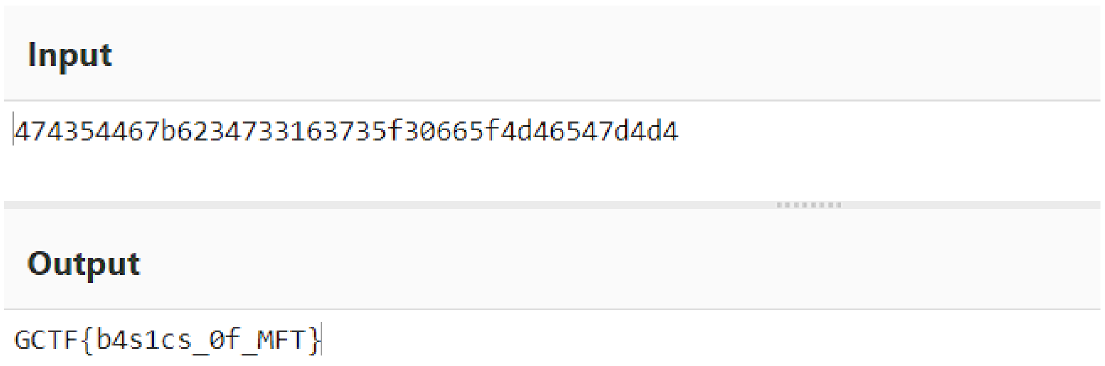
$MFT Browser - C:\Users\60108\Downloads\GCTF2024_Forensics\MFTBasics\$MFT
File
New Range
Find FileID
Cancel

```
$MFT
  [Orphan]
  [Root] .
    $AttrDef
    $BadClus
    $Bitmap
    $Boot
    $Extend
    $LogFile
    $MFT
    $MFTMirr
    $Recycle.Bin
      S-1-5-21-3006358405-3209498723-626421407-
      S-1-5-21-3006358405-3209498723-626421407-
    $Secure
      $Secure
      $Secure
    $UpCase
    $UpCase:$Info
    $Volume
    $WinREAgent
      Scratch
    Documents and Settings
    DumpStack.log.tmp
    pagefile.sys
    PerfLogs
    Program Files
    Program Files (x86)
    ProgramData
    Recovery
    swapfile.sys
    System Volume Information
      IndexerVolumeGuid
      MountPointManagerRemoteDatabase
      tracking.log
      WPSettings.dat
    Users
      All Users
```

```
$MFT
  $MFT Record Nr: 114359, SeqNr: 4
    .\Users\GIC2024\Desktop\flag.txt
    flag.txt
    Parent Directory MFT#: 54439, SeqNr: 3
    Header
      [0x6FADC00] $MFT Record ID: 000400000001BEB7
      [0x6FADC00] Record Offset: 117103616
      [0x00] Signature: FILE
      [0x04] Offset to Update Sequence Array: 48
      [0x06] Number of fix up byte pairs: 3
      [0x08] $LogFile Sequence Number (LSN): 390431365
      [0x10] $MFT Record Sequence Nr: 4
      [0x12] Hard Link Count: 1
      [0x14] Offset to 1st Attribute: 56
      [0x16] Allocation Status: 0x0001
      [0x18] Logical Size of MFT record: 416
      [0x1C] Physical Size of MFT record: 1024
      [0x20] Base Record: 0
      [0x26] Base Record SeqNr: 0
      [0x28] Next Available Attribute ID: 6
      [0x2A] $MFT Record Nr: 114359
      [0x30] Update sequence Number: 5
      [0x30] Update sequence/FixUp Value: 0x0500
      [0x32] FixUp #1: 0x0000
      [0x34] FixUp #2: 0x0000
      [0x1FE] Check Value #1: 0x0500
      [0x3FE] Check Value #2: 0x0500
    Attributes
      [0x038] ID: 00000, Type: (016) 10000000 - $Standa
        [0x03C] Attribute Length: 96
        [0x040] Attribute Non-Resident Status: Resident
        [0x041] Length of Stream Name: 0
        [0x042] Offset to Stream Name: 0
        [0x044] Attribute Flags: 0x0000
        [0x046] Attribute ID: 0
        [0x048] Resident Content Size: 72
        [0x04C] Resident Content Offset: 24
        [0x050] File Created: 05-Sep-2024 18:11:13.7811
        [0x058] File Modified: 05-Sep-2024 18:19:20.092
        [0x060] MFT Record Modified: 05-Sep-2024 18:19:
        [0x068] File Last Accessed: 05-Sep-2024 18:19:2
        [0x070] File Type Flags: 0x00000020
        [0x074] Max Number of Versions: Disabled
        [0x079] Reserve Storage ID: None (00)
```

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   ASCII                             Notes
009  B0 13 D9 01 00 00 00 00 30 00 00 00 70 00 00 00  . . . . . . . . 0 . . . p . . .
00A  00 00 00 00 00 00 04 00 52 00 00 00 18 00 01 00  . . . . . . . . R . . . . . . .
00B  A7 D4 00 00 00 00 03 00 9C E7 65 FE BE FF DA 01  . . . . . . . . . . e . . . . .
00C  9C E7 65 FE BE FF DA 01 9C E7 65 FE BE FF DA 01  . . e . . . . . . . e . . . . .
00D  9C E7 65 FE BE FF DA 01 00 00 00 00 00 00 00 00  . . e . . . . . . . . . . . . .
00E  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00  . . . . . . . .   . . . . . . .
00F  08 03 66 00 6C 00 61 00 67 00 2E 00 74 00 78 00  . . f . l . a . g . . . t . x .
010  74 00 00 00 00 00 00 00 40 00 00 00 28 00 00 00  t . . . . . . . @ . . . ( . . .
011  00 00 00 00 00 00 05 00 10 00 00 00 18 00 00 00  . . . . . . . . . . . . . . . .
012  87 8D 61 04 88 6B EF 11 AB D7 00 0C 29 22 8E 27  . . a . . k . . . . . . ) " . '
013  80 00 00 00 68 00 00 00 00 00 18 00 00 00 01 00  . . . . h . . . . . . . . . . .
014  4C 00 00 00 18 00 00 00 54 68 65 20 73 65 63 72  L . . . . . . . T h e   s e c r
015  65 74 20 63 6F 64 65 20 69 73 3A 20 34 37 20 34  e t   c o d e   i s :   4 7   4
016  33 20 35 34 20 34 36 20 37 62 20 36 32 20 33 34  3   5 4   4 6   7 b   6 2   3 4
017  20 37 33 20 33 31 20 36 33 20 37 33 20 35 66 20    7 3   3 1   6 3   7 3   5 f
018  33 30 20 36 36 20 35 66 20 34 64 20 34 36 20 35  3 0   6 6   5 f   4 d   4 6   5
019  34 20 37 64 34 64 20 34 FF FF FF FF 82 79 47 11  4   7 d 4 d   4 . . . . . y G .
01A  FF FF FF FF 82 79 47 11 00 00 00 00 18 00 00 00  . . . . . y G . . . . . . . . .
01B  FF FF FF FF 82 79 47 11 00 00 00 00 00 00 00 00  . . . . . y G . . . . . . . . .
01C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
01D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
01E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
01F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 00  . . . . . . . . . . . . . . . .
020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
```

Bake that in cyberchef andddd flag captured!



Input

```
474354467b6234733163735f30665f4d46547d4d4d4
```

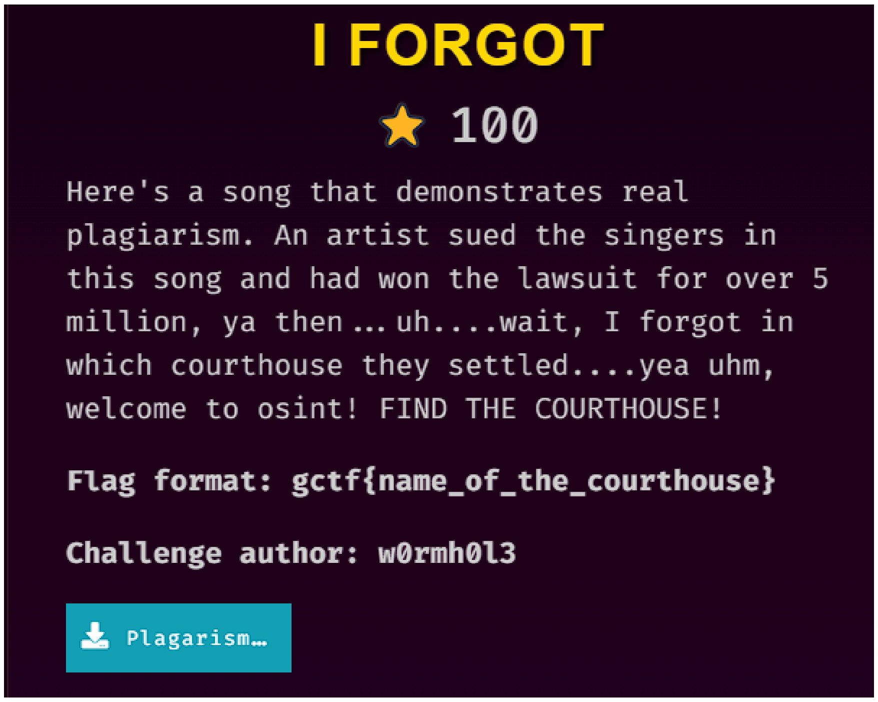
Output

```
GCTF{b4s1cs_0f_MFT}
```

MISC

I FORGOT

Flag: `gctf{United_States_Court_of_Appeals_for_the_Ninth_Circuit}`



I FORGOT

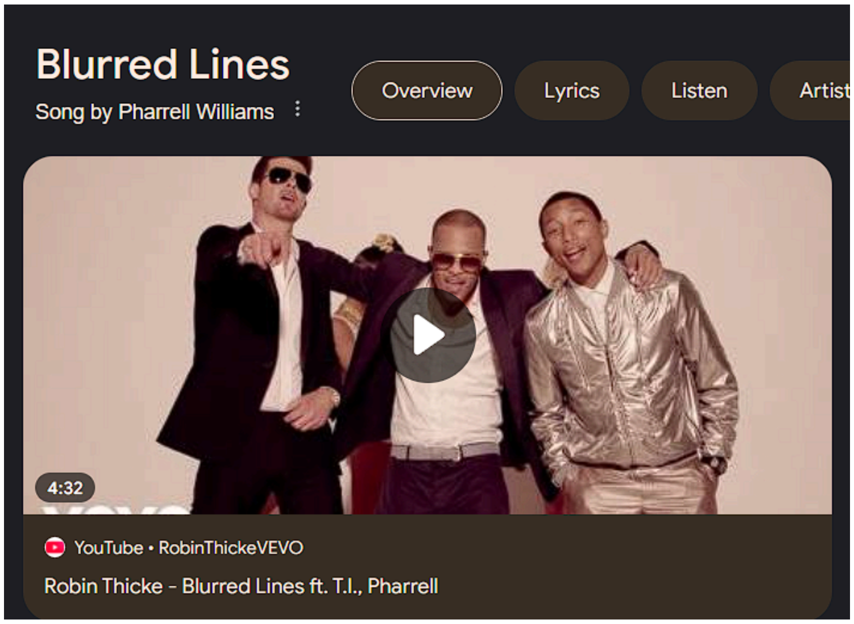
100

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

We were given an MP3 file. Ooh, a song! Since we didn't recognize it, we used Shazam to find out what it was. Now we know the title and who the singer is!



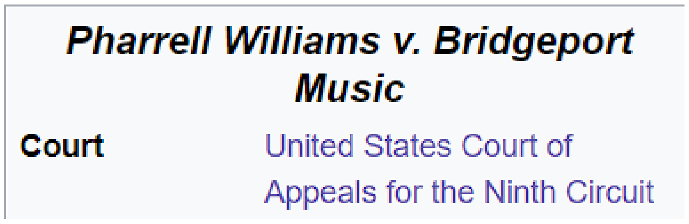
Blurred Lines

Song by Pharrell Williams

YouTube • RobinThickeVEVO

Robin Thicke - Blurred Lines ft. T.I., Pharrell

Next, we Googled "Blurred Lines court trial" and discovered the courthouse mentioned on Wikipedia. And there’s our flag!

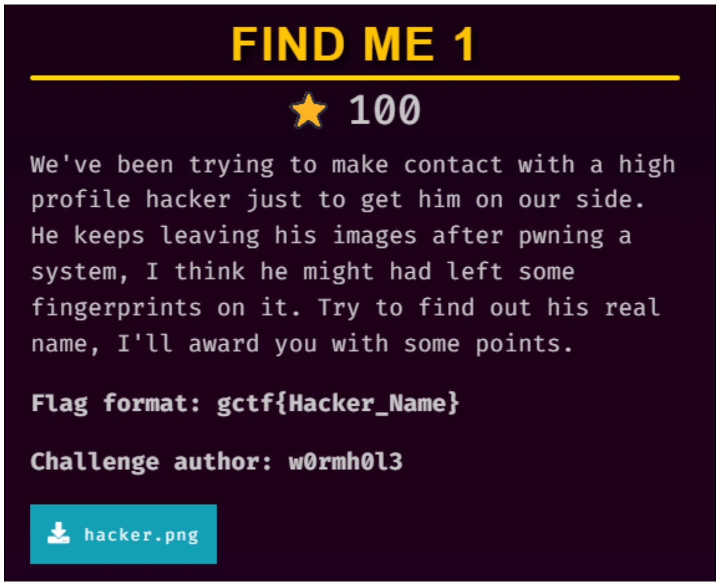


Pharrell Williams v. Bridgeport Music

Court	United States Court of Appeals for the Ninth Circuit

FIND ME 1

Flag: `gctf{Paul_Wildenberg}`



FIND ME 1

100

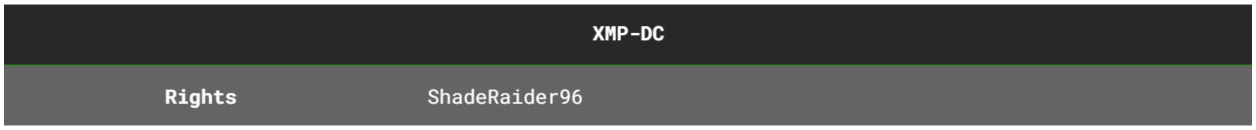
We've been trying to make contact with a high profile hacker just to get him on our side. He keeps leaving his images after pwning a system, I think he might had left some fingerprints on it. Try to find out his real name, I'll award you with some points.

Flag format: gctf{Hacker_Name}

Challenge author: w0rmh0l3

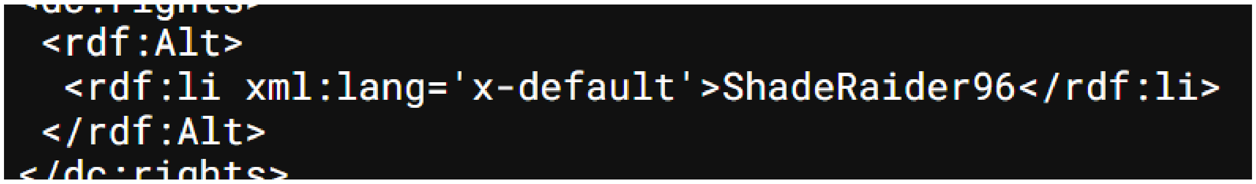
Instead of analyzing the .png file manually, I uploaded it on Aperi'Solve for quick analysis

exiftool:



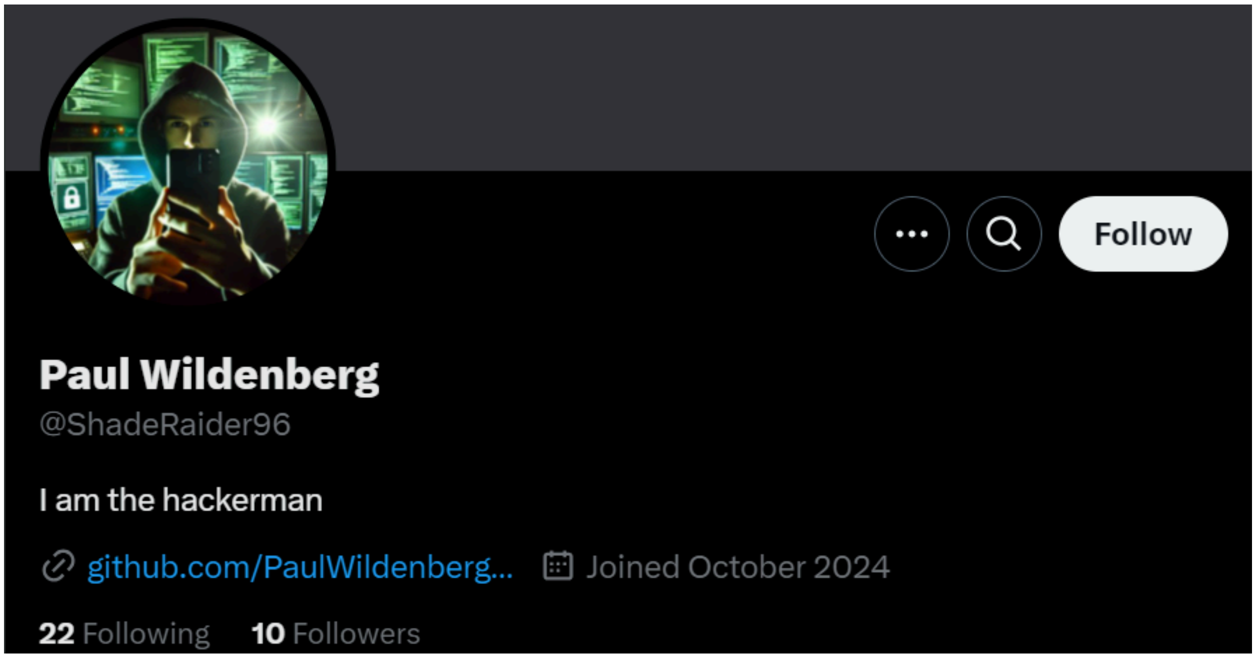
```
XMP-DC
Rights          ShadeRaider96
```

strings:



```
 <rdf:Alt>
  <rdf:li xml:lang='x-default'>ShadeRaider96</rdf:li>
 </rdf:Alt>
```

Results from exiftool and strings lead to this probable username of the hacker. Username found, what else? Find his social media of course! Run the username using OSINT tools like https://instantusername.com/#/ and we found his X account. The flag is his real name, so yeah.



Paul Wildenberg
@ShadeRaider96

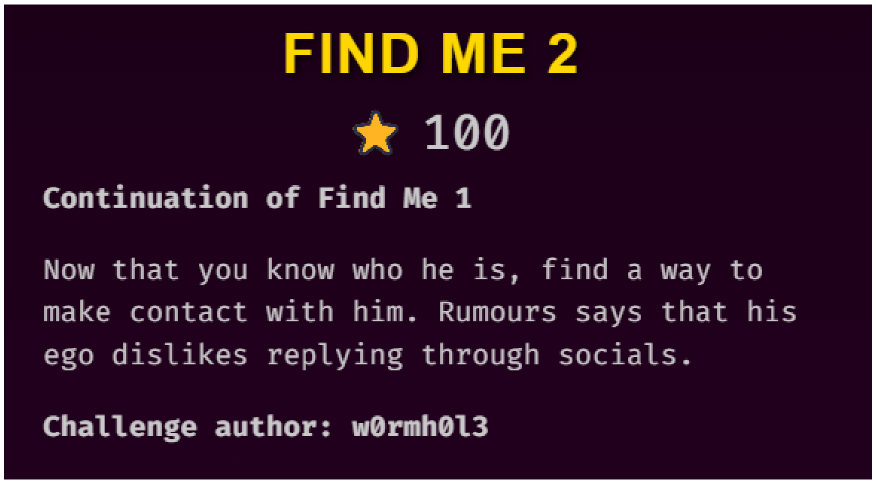
I am the hackerman

github.com/PaulWildenberg... Joined October 2024

22 Following 10 Followers

FIND ME 2

Flag: `gctf{R34L_OS1N7_M4st3r_Hun71ng_T4rg3ts}`

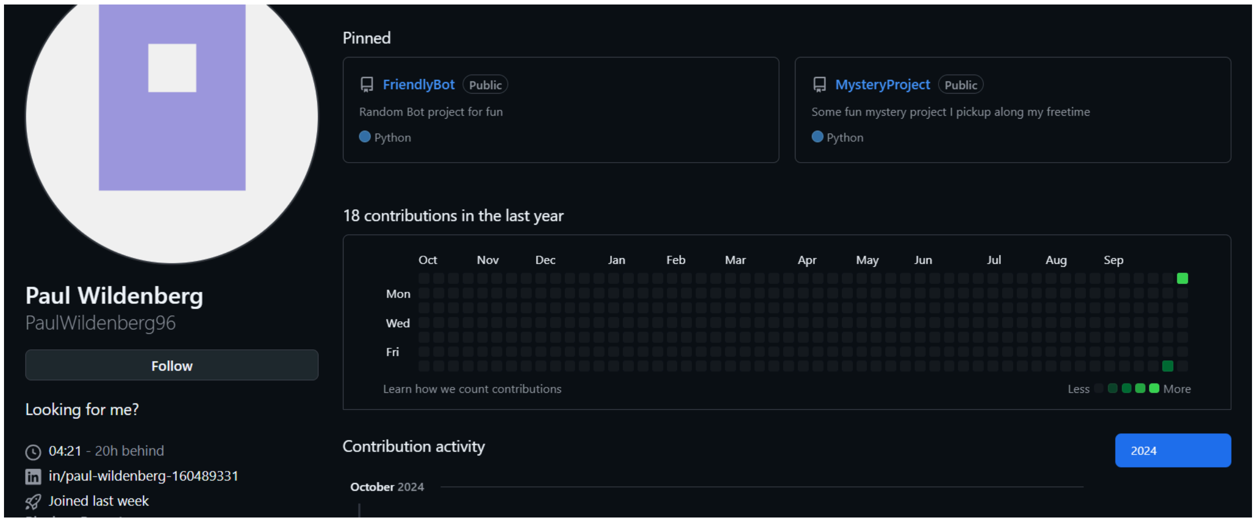

FIND ME 2

100

Continuation of Find Me 1

Now that you know who he is, find a way to make contact with him. Rumours says that his ego dislikes replying through socials.

Challenge author: w0rmh0l3


Paul Wildenberg
PaulWildenberg96

Looking for me?

04:21 - 20h behind
in/paul-wildenberg-160489331
Joined last week

Pinned

FriendlyBot Public
Random Bot project for fun
Python

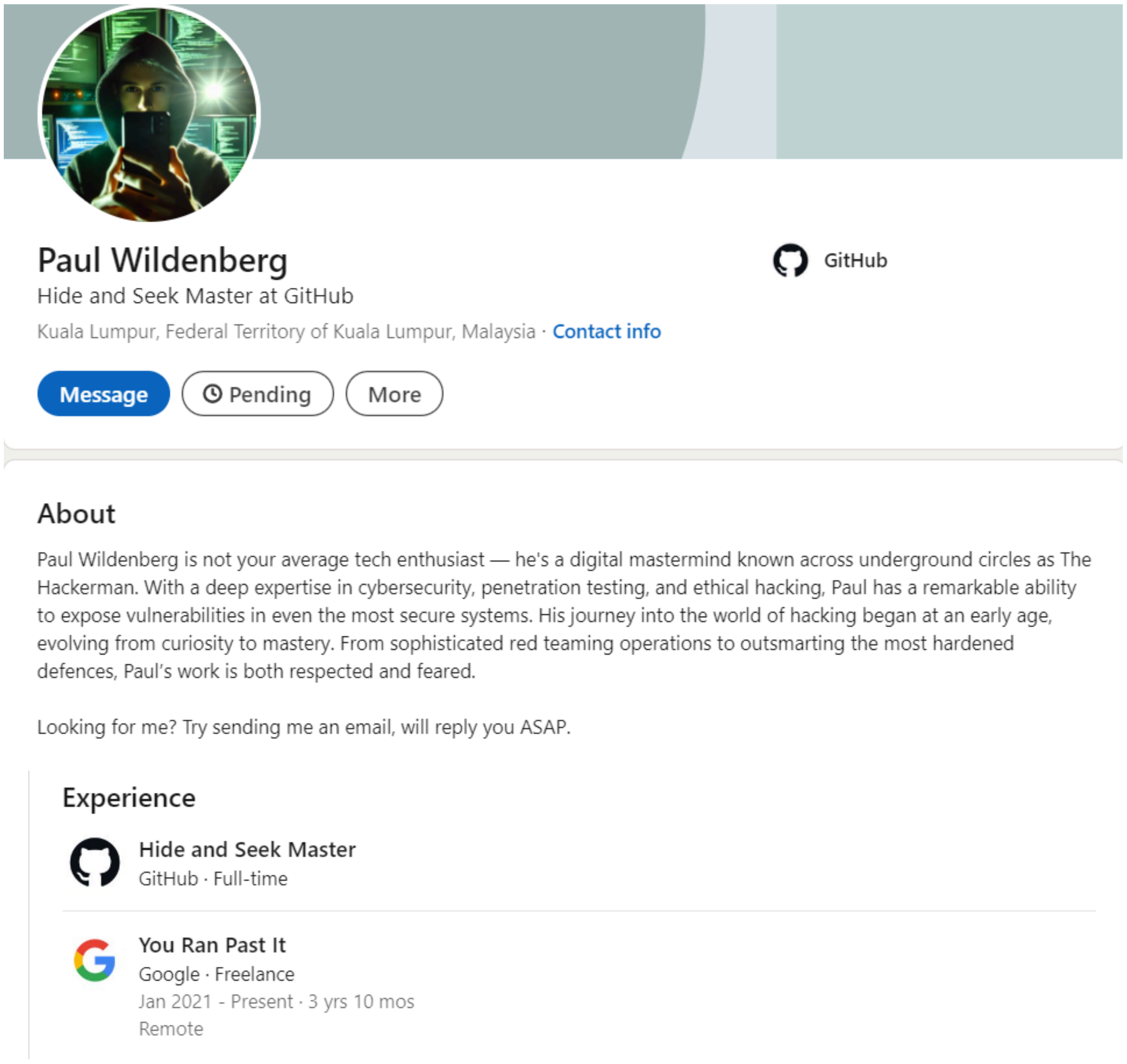
MysteryProject Public
Some fun mystery project I pickup along my freetime
Python

18 contributions in the last year

Contribution activity

October 2024

From the previous X profile, we followed the GitHub link in his bio. However, there was nothing particularly noteworthy in his GitHub repository—just some 'friendly' Python projects he had committed, likely trying to mislead us. Nevertheless, he did include a link to his LinkedIn profile, so we decided to check that out.


Paul Wildenberg

Hide and Seek Master at GitHub

Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia · Contact info

About

Paul Wildenberg is not your average tech enthusiast — he's a digital mastermind known across underground circles as The Hackerman. With a deep expertise in cybersecurity, penetration testing, and ethical hacking, Paul has a remarkable ability to expose vulnerabilities in even the most secure systems. His journey into the world of hacking began at an early age, evolving from curiosity to mastery. From sophisticated red teaming operations to outsmarting the most hardened defences, Paul's work is both respected and feared.

Looking for me? Try sending me an email, will reply you ASAP.

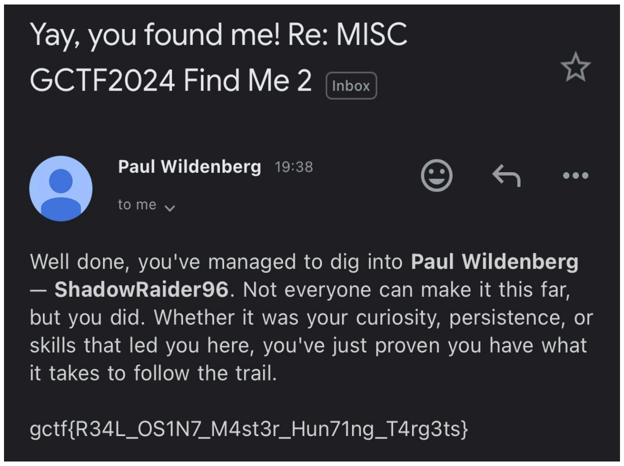
Experience

Hide and Seek Master
GitHub · Full-time

You Ran Past It
Google · Freelance
Jan 2021 - Present · 3 yrs 10 mos
Remote

He mentioned that we could try sending him an email, and we believe the answer might be there since Gmail isn't a social media platform. We used the username from X, which is paulwildenberg96, as his email because, well, people often do that (sometimes we can be a bit forgetful, like Dory xD). So his complete email address is paulwildenberg96@gmail.com.

And we tried sending an email. He replied ASAP which is surprising hehe. But yeah, we got the flag.


Yay, you found me! Re: MISC GCTF2024 Find Me 2

Paul Wildenberg 19:38

to me

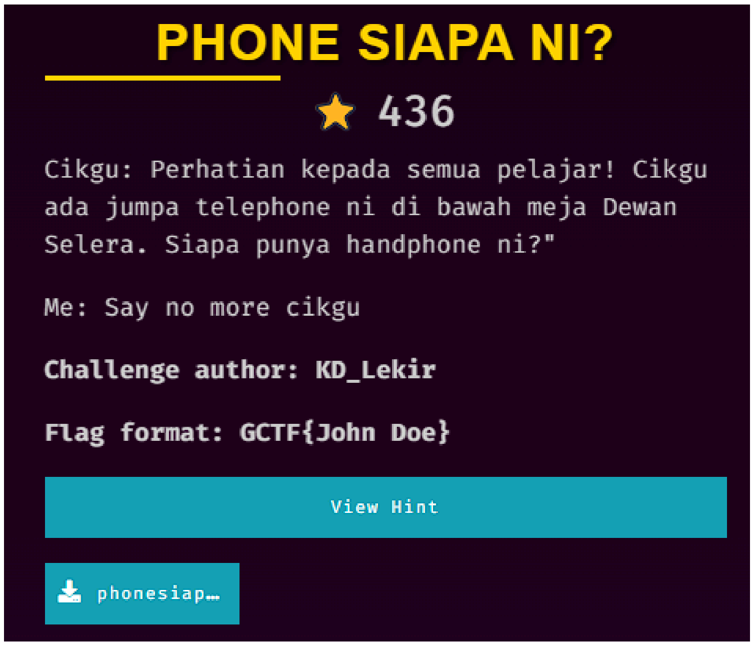
Well done, you've managed to dig into **Paul Wildenberg — ShadowRaider96**. Not everyone can make it this far, but you did. Whether it was your curiosity, persistence, or skills that led you here, you've just proven you have what it takes to follow the trail.

gctf{R34L_OS1N7_M4st3r_Hun71ng_T4rg3ts}

RE

PHONE SIAPA NI?

Flag: `GCTF{Liyana Nura binti Fakhri}`



PHONE SIAPA NI?

436

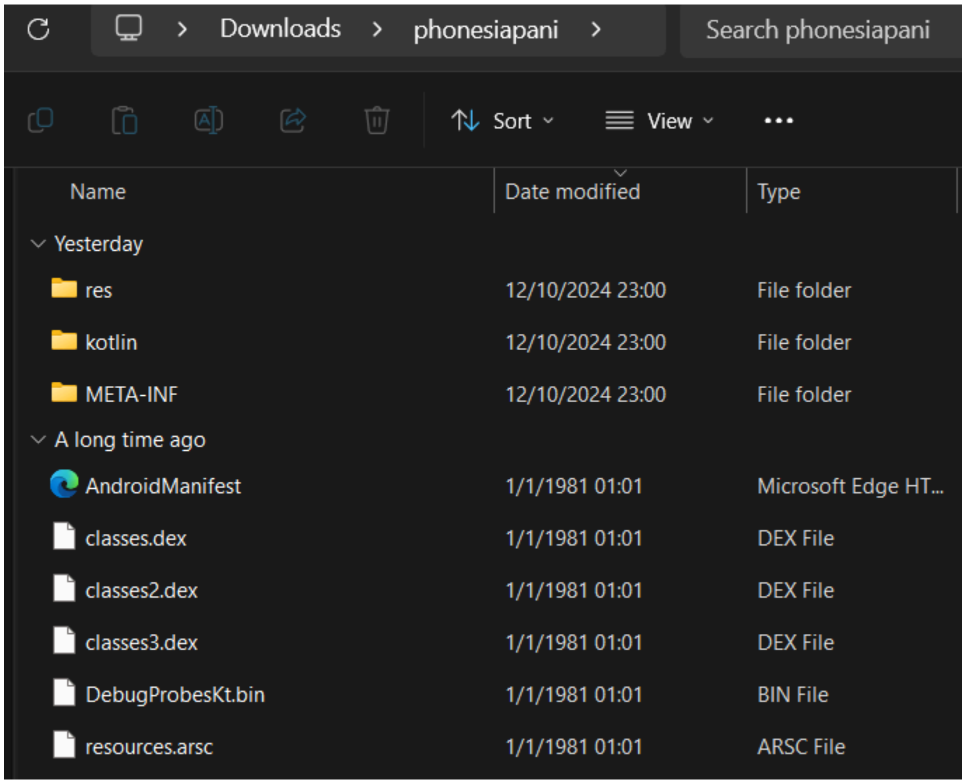
Cikgu: Perhatian kepada semua pelajar! Cikgu ada jumpa telephone ni di bawah meja Dewan Selera. Siapa punya handphone ni?"

Me: Say no more cikgu

Challenge author: KD_Lekir

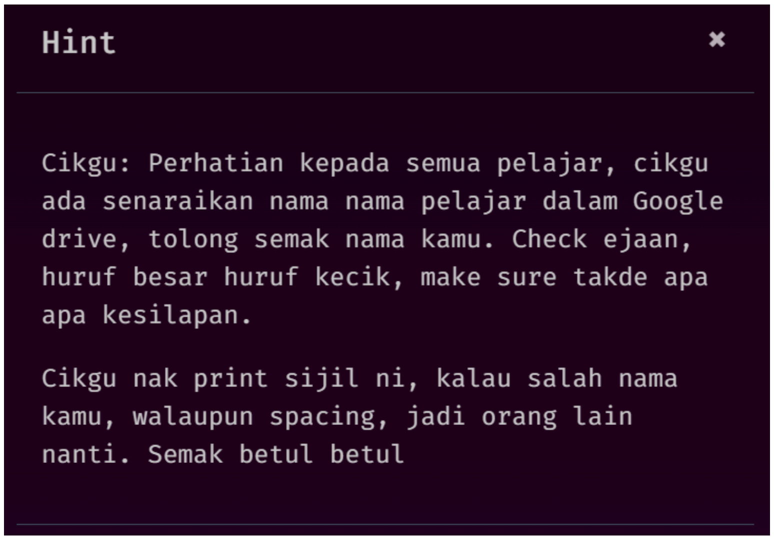
Flag format: GCTF{John Doe}

We received a zip file with various files and folders. This is the typical structure of an APK file.



Name	Date modified	Type
Yesterday		
res	12/10/2024 23:00	File folder
kotlin	12/10/2024 23:00	File folder
META-INF	12/10/2024 23:00	File folder
A long time ago		
AndroidManifest	1/1/1981 01:01	Microsoft Edge HT...
classes.dex	1/1/1981 01:01	DEX File
classes2.dex	1/1/1981 01:01	DEX File
classes3.dex	1/1/1981 01:01	DEX File
DebugProbesKt.bin	1/1/1981 01:01	BIN File
resources.arsc	1/1/1981 01:01	ARSC File

At first I thought we needed to run the app through an android emulator. But since the challenge's hint features a teacher's dialogue, we suspected that the files we needed to analyze were the class files in .dex format.

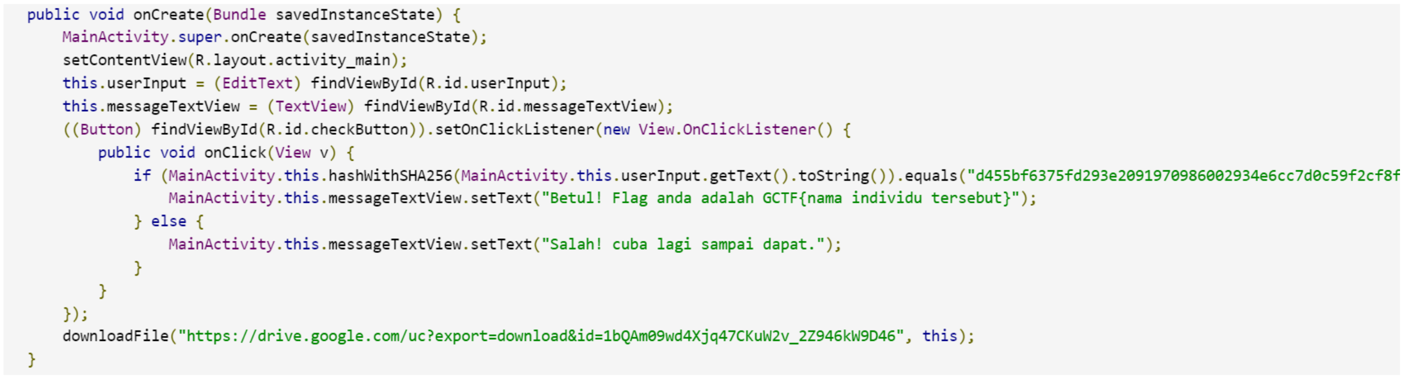


Hint

Cikgu: Perhatian kepada semua pelajar, cikgu ada senaraikan nama nama pelajar dalam Google drive, tolong semak nama kamu. Check ejaan, huruf besar huruf kecik, make sure takde apa apa kesilapan.

Cikgu nak print sijil ni, kalau salah nama kamu, walaupun spacing, jadi orang lain nanti. Semak betul betul

We opened the file using https://www.decompiler.com/. We looked for clues and discovered some valuable hints in the classes3.dex file, as shown below:



```
public void onCreate(Bundle savedInstanceState) {
    MainActivity.super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    this.userInput = (EditText) findViewById(R.id.userInput);
    this.messageTextView = (TextView) findViewById(R.id.messageTextView);
    ((Button) findViewById(R.id.checkButton)).setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            if (MainActivity.this.hashWithSHA256(MainActivity.this.userInput.getText().toString()).equals("d455bf6375fd293e2091970986002934e6cc7d0c59f2cf8f
                MainActivity.this.messageTextView.setText("Betul! Flag anda adalah GCTF{nama individu tersebut}");
            } else {
                MainActivity.this.messageTextView.setText("Salah! cuba lagi sampai dapat.");
            }
        }
    });
    downloadFile("https://drive.google.com/uc?export=download&id=1bQAm09wd4Xjq47CKuW2v_2Z946kW9D46", this);
}
```

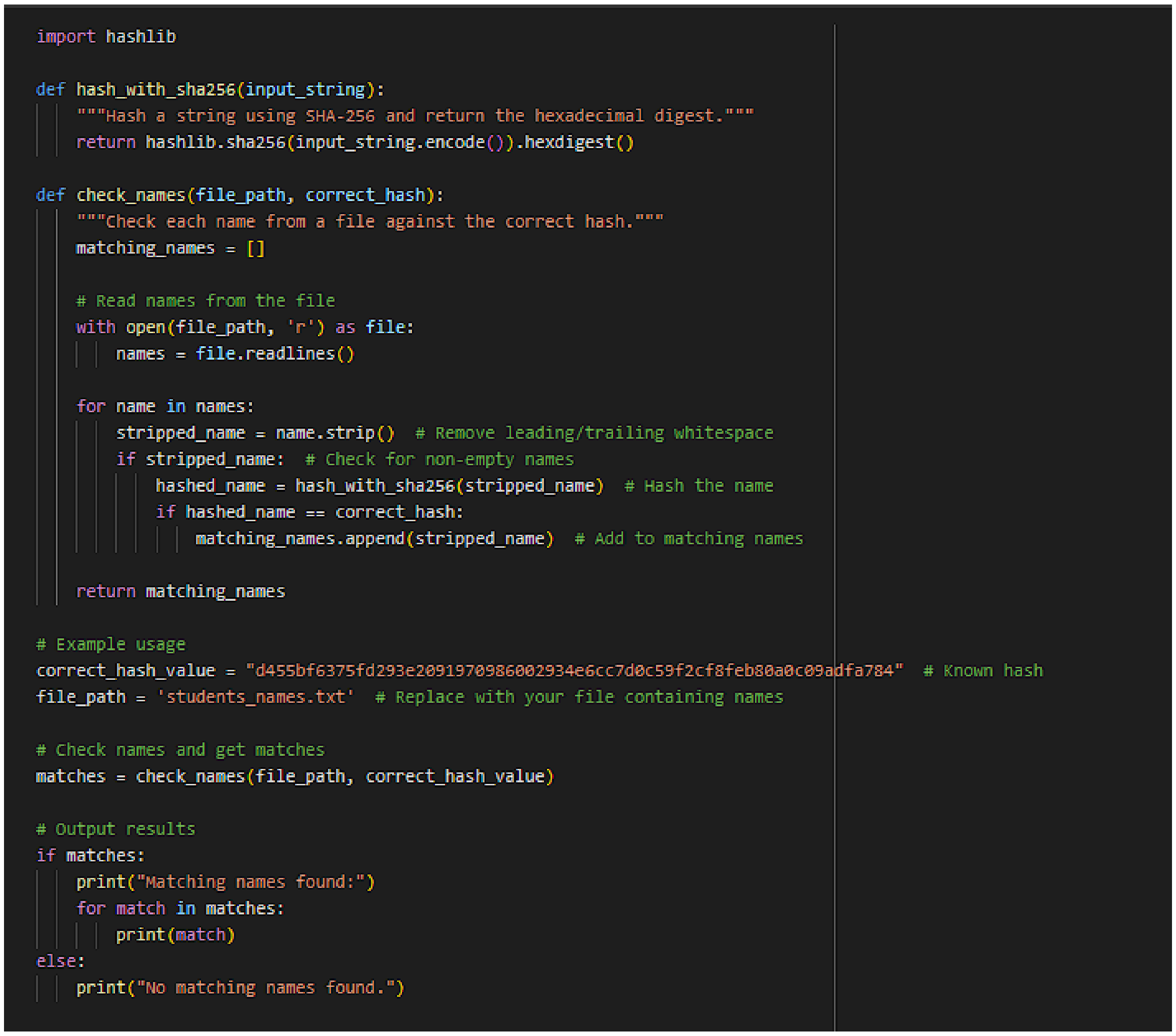
The full code is for an Android app that takes user input, checks if it matches a specific hashed value (a secret flag), and downloads a file from the internet. It displays messages to the user based on their input. What we want to focus on is the method mentioning the flag 'GCTF{}'. Basically when the user clicks the button, the app checks if the hashed version of their input matches a specific hashed value. If it matches, a success message is shown (flag); otherwise, an error message is displayed.

We copy pasted the Google Drive link on the browser and it auto-downloads this text file which contains a list of 50,000 names :0

senaraiPelajarUITM.txt

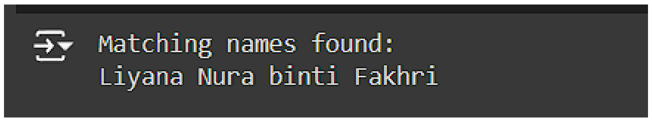
Since the flag success message is pre-written output, meaning there's no flag value from there, so what we need to do is reverse the method! Input a name that matches the hash value so we can get the success message. Technically speaking, it's a pain to input 50k names one by one duh, so of course we need to use a script to automate this.

We sought help from chatgpt to create a code to compare the hash value with the list of names. The code is as follows:



```python
import hashlib

def hash_with_sha256(input_string):
    """Hash a string using SHA-256 and return the hexadecimal digest."""
    return hashlib.sha256(input_string.encode()).hexdigest()

def check_names(file_path, correct_hash):
    """Check each name from a file against the correct hash."""
    matching_names = []

    # Read names from the file
    with open(file_path, 'r') as file:
        names = file.readlines()

    for name in names:
        stripped_name = name.strip()  # Remove leading/trailing whitespace
        if stripped_name:  # Check for non-empty names
            hashed_name = hash_with_sha256(stripped_name)  # Hash the name
            if hashed_name == correct_hash:
                matching_names.append(stripped_name)  # Add to matching names

    return matching_names

# Example usage
correct_hash_value = "d455bf6375fd293e2091970986002934e6cc7d0c59f2cf8feb80a0c09adfa784"  # Known hash
file_path = 'students_names.txt'  # Replace with your file containing names

# Check names and get matches
matches = check_names(file_path, correct_hash_value)

# Output results
if matches:
    print("Matching names found:")
    for match in matches:
        print(match)
else:
    print("No matching names found.")
```

And we got this name. That's our flag.



```
Matching names found:
Liyana Nura binti Fakhri
```