

Write Up Kashi CTF 2025

ICC Pisang Epe - pengen banget lolos gelatik plss
tapi klo gabisa yaudahlah coba tahun depan lagi :)



Disusun oleh:

- .nadhif - Muh. Nadhiftamma Ayatilla A.P
 - chell07 - Chelsea Elysia Chandean
- giginaga - Andi Ghaniyatera Febriana Harfa Makkasau

Crypto

1. Lost Frequencies

The screenshot shows a challenge interface with the following details:

- Challenge:** Lost Frequencies
- Points:** 100
- Author:** Argus817
- Description:** Zeroes, ones, dots and dashes
Data streams in bright flashes
- Morse Code Example:** 111 0000 10 111 1000 00 10 01 010 1011 11 111 010
000 0
- Note:** NOTE: Wrap the capitalized flag in KashiCTF{ }
- Buttons:** Flag, Submit

Pada soal ini kami diberi beberapa string angka yang terdiri dari 1 dan 0, setelah kami selidiki ternyata angka-angka tersebut merujuk pada sandi morse. berikut langkah untuk mendapatkan flag:

1. Menganalisis Pola Biner

Pertama, kita perhatikan bahwa angka biner ini terdiri dari kombinasi **1** dan **0**. Karena challenge menyebutkan "Zeroes, ones, dots, and dashes" (nol, satu, titik, dan garis), ini mengarahkan kita ke **kode Morse**, di mana:

- **1** bisa mewakili **garis (-)**,
- **0** bisa mewakili **titik (.)**.

2. Mengubah Biner ke Morse

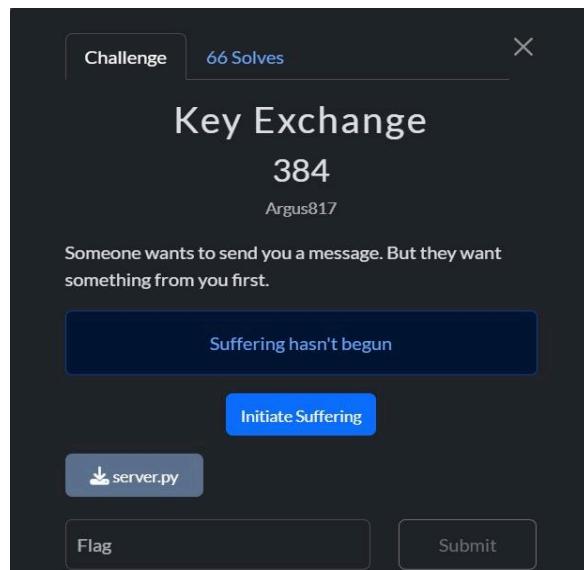
Kita akan mengubah setiap kelompok biner menjadi simbol Morse:

- 111 → --- (O)
- 0000 → (H)
- 10 → - (N)
- 111 → --- (O)
- 1000 → -... (B)
- 00 → .. (I)
- 10 → - (N)
- 01 → .- (A)
- 010 → .-. (R)

- $1011 \rightarrow -\cdot-$ (Y)
- $11 \rightarrow \cdot\cdot$ (M)
- $111 \rightarrow \cdot\cdot\cdot$ (O)
- $010 \rightarrow \cdot\cdot$ (R)
- $000 \rightarrow \dots$ (S)
- $0 \rightarrow \cdot$ (E)

flag: KashiCTF{OHNOBINARY_MORSE}

2. Key Exchange



Pada soal ini kami diberikan sebuah nc dengan servernya dalam bentuk python, ketika di run dalam Ubuntu, rupanya ini sepertinya menggunakan Diffie Hellman key exchange, lalu server meminta nilai Public key-x dan y koord. sehingga untuk mempermudah, kami menggunakan ChatGPT dan mendapatkan koord public key x dan y.

Sepertinya server memeriksa validitas titik yang dikirim, karena setelah kita mengirim $(0,0)$, server langsung keluar dengan "OK Bye!! 😊" tanpa memberikan ciphertext.

Kita bisa coba pendekatan lain:

Strategi Lanjutan

1. Gunakan Titik Generator G

- Jika server tidak memeriksa apakah titik yang kita kirim adalah titik generator, kita bisa mencoba mengirim koordinat G yang sudah diberikan dalam kode:

```
vbnet
Public Key x-coord: 26247035095799689268623156744566981891852923491109213387815615900
Public Key y-coord: 83257109614890299855467512895201081792878530488613155947092059024
```

Jika berhasil, shared secret akan menjadi $n_A * G$, yang merupakan public key Alice (P_A).

Setelah mendapatkan nilai public key x & y coord. kami menginput nilainya dan keluarlah message dengan chipertext. Lalu kami bertanya lagi pada ChatGPT dan diberikanlah sebuah code python untuk men-dekripsi chipertext tersebut.

```

chelsea07@chels77: $ nc kashictf.iitbhucybersec.in 62601
Received from Weierstrass:
    Here are the curve parameters (NIST P-384)
    p = 39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319
    a = -3
    b = 27580193559959765877849011840389904809385690586361568521428707301988689241389860865136260764883745107765439761236575
    And my Public Key: (32765930154653112179512710833156730953778336866050492019100655185067730887639831274035392853275923755572323123865229, 1066
50267467418028844153178013580641736537878772512062915085753554161439407732414442018241424193181416621745640997)

Send to Weierstrass:
    Public Key x-coord: 2624703509579968926862315674456698189185292349110921338781561590092551885473805008902388053975719786650872476732087
    Public Key y-coord: 832571096189029985467512895201081792878530486131559470928590248050319984419224438643760392947333078886511627871

Received from Weierstrass:
    Message: {"iv": "14bc0884df7733c827b64c8b2450d330", "ciphertext": "d1eb697b03baa42a2d4f83118fc2af96b01818f6f500e3781c3a04c545d72de0b79cc522e1
55550ce576f59850cd8ae139cd257d4a0981136b5ad8453cb1db35d71025eb521fb87bae16bb52b4992906ef22531c5b6949af9be0b82228e6871"}

ChatGPT ^
AES Decryption Example ▾

1   from Crypto.Cipher import AES
2   from Crypto.Util.Padding import unpad
3   import hashlib
4   import json
5
6   # Data dari server
7   data = {
8       "iv": "14bc0884df7733c827b64c8b2450d330",
9       "ciphertext": "d1eb697b03baa42a2d4f83118fc2af96b01818f6f500e3781c3a04c545d72de0b79cc522e1
10      55550ce576f59850cd8ae139cd257d4a0981136b5ad8453cb1db35d71025eb521fb87bae16bb52b4992906ef22531c5b6949af9be0b82228e6871"
11
12      # Hitung AES Key
13      shared_secret = 327659301546531121795127108331567309537783368060504920191006551850677308876398312740353928532759237
14      sha1 = hashlib.sha1()
15      sha1.update(str(shared_secret).encode("ascii"))
16      key = sha1.digest()[:16]
17
18      # Konversi IV dan Ciphertext ke bytes
19      iv = bytes.fromhex(data["iv"])
20      ciphertext = bytes.fromhex(data["ciphertext"])
21
22      # Dekripsi
23      cipher = AES.new(key, AES.MODE_CBC, iv)
24      plaintext = unpad(cipher.decrypt(ciphertext), 16)
25
26      print("FLAG:", plaintext.decode())
27


```

ng kita harus memberikan
tan y untuk Public Key kita
isa mencoba beberapa strategi
cription Example

Lalu setelah code python tersebut di run dalam Ubuntu, kami mendapatkan flag terenkripsi.

```

See /usr/share/doc/python3.12/README.venv for more information.

note: If you believe this is a mistake, please contact your Python installation or OS distribution provider. You can override this
of breaking your Python installation or OS, by passing --break-system-packages.

hint: See PEP 668 for the detailed specification.

(venv) chelsea07@chels77: $ nano exc.py
(venv) chelsea07@chels77: $ python3 exc.py
Traceback (most recent call last):
  File "/home/chelsea07/exc.py", line 1, in <module>
    from Crypto.Cipher import AES
ModuleNotFoundError: No module named 'Crypto'
(venv) chelsea07@chels77: $ ^C
(venv) chelsea07@chels77: $ python3 -m venv myenv
(venv) chelsea07@chels77: $ source myenv/bin/activate
(myenv) chelsea07@chels77: $ source myenv/bin/activate
(myenv) chelsea07@chels77: $ pip install pycryptodome
Requirement already satisfied: pycryptodome in ./myenv/lib/python3.12/site-packages (3.21.0)
(myenv) chelsea07@chels77: $ nano exc.py
(myenv) chelsea07@chels77: $ python3 exc.py
FLAG: NaeusGRX{L_r3H3Nv3h_kq_Sun1Vm_03w_4fg_4lx_1_t0d_a4q_lk1s_X0hcc_Dd4J_zNAYrN1A}

Hint: DamnKeys
(myenv) chelsea07@chels77: $ 

```

Setelah menganalisis flag yang ter-enkripsi kami mencoba menggunakan Vignere chiper dan rupanya benar, flagnya pun kami dapatkan.



flag:

KashiCTF{I_r3V3Al3d_my_Pub1Ic_K3y_4nd_4ll_1_g0t_w4s_th1s_L0usy_Fl4G_zBNOnP1I}

MISC

1. Easy Jail



Pada soal ini kami diberikan server nc dengan zip challenge yang berisi chall python, docker dan fake flag. Lalu, kami meminta bantuan ChatGPT dan kami diminta memasukkan beberapa perintah untuk di-input pada server dalam Ubuntu.

2. Access Built-in Functions
If the previous command works, try:

```
markdown
```

```
().__class__.__base__.__subclasses__()
```

This lists all available subclasses in Python. We can use it to find useful functions like `os.system`.

3. Extract the Flag
Once you have the list of subclasses, find `os.system` or `open()` using:

```
java
```

```
[x for x in ()).__class__.__base__.__subclasses__() if 'file' in str(x).lower()]
```

If you find a file-related class, you can try:

```
lua
```

```
open('/flag.txt').read()
```

Lalu setelah diinput, Flag-nya pun kami dapatkan.

```
>> ^C
chelsea07@chels77: ~
>> nc kashictf.iitbhcybersec.in 54472
[...]
>> O.__class__.__base__
O.__class__.__base__ --> <class 'object'>
>> O.__class__.__base__.__subclasses__()
O.__class__.__base__.__subclasses__() --> [<class 'type'>, <class 'async_generator'>, <class 'bytearray_iterator'>, <class 'bytearray'>, <class 'bytes_method'>, <class 'callable_iterator'>, <class 'PyCapsule'>, <class 'cell'>, <class 'classmethod_descriptor'>, <class 'classmethod'>, <class 'code'>, <class 'contextvars.ContextVar'>, <class 'contextvars.Context'>, <class 'coroutine'>, <class 'dict_items'>, <class 'dict_itemiterator'>, <class 'dict_keyiterator'>, <class 'mappingproxy'>, <class 'dict_reverseitemiterator'>, <class 'dict_reversekeyiterator'>, <class 'dict_reversevalueiterator'>, <class 'dict_values'>, <class 'filter'>, <class 'float'>, <class 'frame'>, <class 'frozenset'>, <class 'function'>, <class 'generator'>, <class 'getset_descriptor'>, <class 'list_reverseiterator'>, <class 'list'>, <class 'longrange_iterator'>, <class 'int'>, <class 'map'>, <class 'member_descriptor'>, <class 'memoryview'>, <class 'udef'>, <class 'module'>, <class 'odict_iterator'>, <class 'pickle.PickleBuffer'>, <class 'property'>, <class 'range_iterator'>, <class 'range'>, <class 'set_iterator'>, <class 'slice'>, <class 'statimethod'>, <class 'stderpprinter'>, <class 'super'>, <class 'traceback'>, <class 'str'>, <class 'wrapper_descriptor'>, <class 'zip'>, <class 'types.GenericAlias'>, <class 'anext_awaitable'>, <class 'async_generator_aseerator_wrapped_value'>, <class '_buffer_wrapper'>, <class 'Token.MISSING'>, <class 'coroutine_wrapper'>, <class 'generic_alias_iterator'>, <class 'item_y_node'>, <class 'hamt_bitmap_node'>, <class 'hamt_collision_node'>, <class 'hamt'>, <class 'sys._legacy_event_handler'>, <class 'InterpreterID'>, <class 'ry_iterator'>, <class 'method-wrapper'>, <class 'types.SimpleNamespace'>, <class 'NoneType'>, <class 'NotImplementedType'>, <class 'positions_iterator'>, <class 'weakref.CallableProxyType'>, <class 'weakref.ProxyType'>, <class 'weakref.ReferenceType'>, <class 'typing.TypeAliasType'>, <class 'typing.GeneratorType'>, <class 'typing.ParamSpec'>, <class 'typing.ParamSpecArgs'>, <class 'typing.ParamSpecKwargs'>, <class 'EncodingMap'>, <class 'fieldnameiterator'>, <class '_frozen_importlib._WeakValueDictionary'>, <class '_frozen_importlib._BlockingOnManager'>, <class '_frozen_importlib._ModuleLockManager'>, <class '_frozen_importlib.ModuleSpec'>, <class '_frozen_importlib.BuiltinImporter'>, <class '_frozen_importlib.FrozenImporter'>, <class '_thread.Lock'>, <class '_thread.RLock'>, <class '_thread.local'>, <class '_io.IncrementalNewlineDecoder'>, <class '_i.ScandirIterator'>, <class 'posix.DirEntry'>, <class '_frozen_importlib_external.WindowsRegistryFinder'>, <class '_frozen_importlib_external.LoaderBase'
```

flag: KashiCTF{3V4L_41NT_54F3_0Q6tzIaZ}

2. SNOWy Evening



Pada challange ini kami diberi sebuah file txt yang berisi sebuah puisi, setelah kami teliti dan sesuai dengan petunjuk pada judul challange ternyata puisi tersebut dapat dienkripsi menggunakan tools snow pada linux. berikut ini adalah file txt nya:

```
Pity, in place of love,  
That pettiest of gifts,  
Is but a sugar-coating over neglect.  
Any passerby can make a gift of it  
To a street beggar,  
Only to forget the moment the first corner is turned.  
I had not hoped for anything more than day.
```

You left during the last watch of night.

```
I had hoped you would say goodbye,  
Just say 'Adieu' before going away,  
What you had said another day,  
What I shall never hear again.
```

```
In their place, just that one word,  
Bound by the thin fabric of a little compassion  
Would even that have been too much for you to bear?
```

When I first awoke from sleep
My heart fluttered with fear
Lest the time had been over.
I rushed out of bed.
The distant church clock chimed half past twelve
I sat waiting near the door of my room
Resting my head against it,
Facing the porch through which you would come out

Langkah untuk mendapatkan flag:

1. Menggunakan tools “snow” pada linux

dengan menggunakan password “Aakash”, kita bisa menggunakan tools snow dengan menginput command berikut:

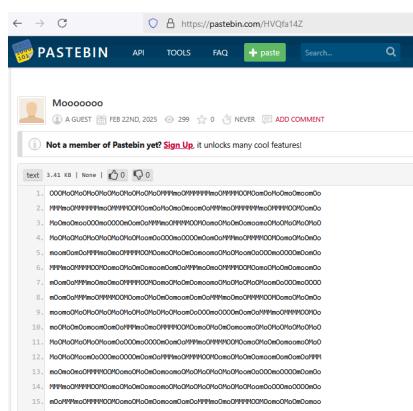
```
./snow -C -p Aakash /directory/to/poemmm.txt
```

```
nadhif@ubuntu:~/Downloads/snow$ find / -name "poem.txt" 2>/dev/null  
nadhif@ubuntu:~/Downloads/snow$ ./snow -C -p Aakash /home/nadhif/D  
https://pastebin.com/HVQfa14Znadhif@ubuntu:~/Downloads/snow$ ./sno  
https://pastebin.com/HVQfa14Znadhif@ubuntu:~/Downloads/snow$ ./sno  
https://pastebin.com/HVQfa14Znadhif@ubuntu:~/Downloads/snow$ ./sno
```

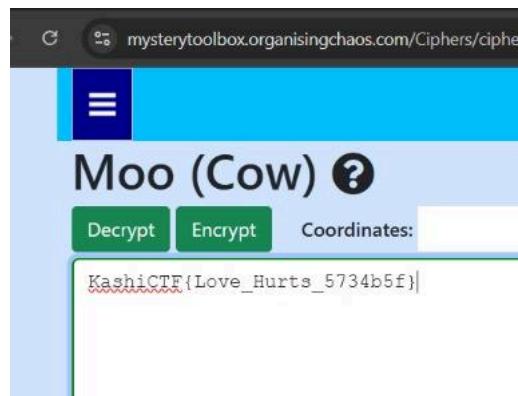
dengan melakukan command terebut kita akan mendapatkan sebuah link:
<https://pastebin.com/HVQfa14Z>

2. Dekripsi MOO

Melalui link tersebut kami mendapatkan sebuah teks yang terdiri dari karakter M dan O yang sangat panjang,



dengan menggunakan Cow Cipher kita akhirnya bisa mendapatkan flagnya



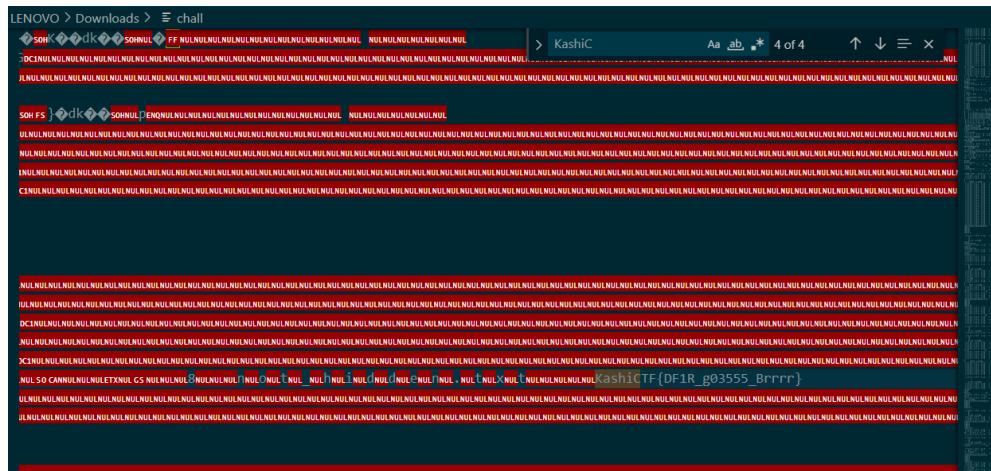
flag: KashiCTF{Love_Hurts_5734b5f} *upsolve

Forensics

1. Memories Bring Back You



Pada challenge ini diberikan sebuah file dengan ukuran yang cukup besar yaitu sekitar 1GB lalu ketik kami memncoba membuka di notepad terjadi lag yang cukup parah karena ukuranfile yang besar. Selanjutnya, kami mencoba menggunakan VS Code dan ternyata berhasil, di dalam file tersebut terdapat sangat banyak elemen dan kata, untuk mempermudah kami menggunakan Ctrl+F untuk menemukan format flag yang kami cari dalam file tersebut, setelah menemukan beberapa fake flag, akhirnya kami mendapatkan flag aslinya.



flag: KashiCTF{DF1R_g03555_Brrrr}

2. Restaurant



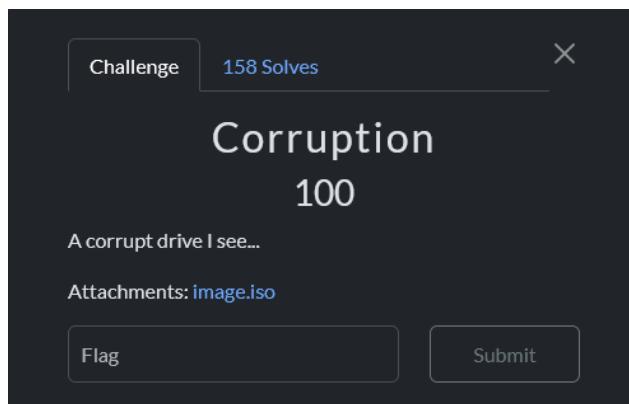
Pada challenge ini kami diberikan sebuah file png yang berisi gambar pasta, lalu setelah mencoba beberapa tools kami menemukan sedikit kejanggalan saat menggunakan hexed.it karena terdapat bit dengan pola a dan b di bagian akhir bitnya.

```
0000B5B0 DB D6 2F D2 B6 7F DB 47 95 F6 AF 77 87 86 DC B2 0000B5C0 0B FF 00 33 A1 FB 68 36 1A 5D 19 28 29 6A 75 E0 0000B5D0 23 A4 48 46 1A D7 DA 42 9C 00 D4 ED 72 C3 2F F6 0000B5E0 D5 91 F4 8D 17 DB 44 4D 56 D2 AC 32 FF 08 6D 59 0000B5F0 5F 49 51 7D B4 12 49 03 01 60 1B 7D CA 34 B4 2D 0000B600 32 53 86 21 D4 50 78 88 A0 42 D0 DA 8D 8A 03 A6 0000B610 D5 B3 8B D5 AE A4 2F AA AC 05 5B 1B 4E A0 57 82 0000B618 55 -1 0000B628 D1 A1 1E 2F FD 9A FC F2 81 DC 1E 77 0B 76 0000B630 DB 92 3E B7 EF 39 27 D1 A7 A8 BC 3E CF 17 74 70 0000B640 ED 81 ZA FA C5 14 D4 22 27 7A BC DC 3E EF 5F 0000B650 5C 08 95 F0 B3 9D FF 00 88 D0 93 BF F8 AA 7E E9 0000B658 8F 7F 11 E5 DD 1E A7 7C 6C EF F7 8D 0F B5 55 4B 0000B660 45 16 F1 8C C6 35 19 11 5D D6 F1 8C C6 35 19 11 0000B668 E1 EE 9E 1D A5 D5 19 B9 AF E8 B5 95 A3 9A E4 48 0000B670 C8 C8 76 93 2C 8C 8C 88 A2 C8 C8 C8 8A 0000B678 2C 8C 88 A2 C8 DE 31 A8 C8 BA 2D E3 19 8C 0000B680 6A 32 22 97 5B C6 33 18 D4 64 45 16 F1 8C C6 35 0000B688 19 11 4B AD E3 19 8C 6A 32 22 97 59 1B C6 35 19 0000B6C0 11 4B AD E3 19 8C 6A 32 22 97 5B C6 33 18 D4 64 0000B6D0 45 16 F1 8C C6 35 19 11 5D D6 F1 8C C6 35 19 11 0000B6E0 4B AD E3 19 8C 6A 32 22 97 5B C6 33 18 D4 64 45 0000B6F0 2B 87 BC 66 31 AB C8 A8 BE 87 BC 66 31 AB C8 8A 0000B700 5D 6F 18 CC 63 51 91 14 5B C6 35 19 19 11 45 BC 0000B710 63 31 8D 46 44 52 EB 78 C6 63 1A 8C 88 AE EB 78 0000B720 C6 63 1A 8C 88 A5 D6 F1 8D 46 46 44 54 B7 8C 6A 0000B730 32 32 2B 88 78 C6 A3 23 22 28 B2 32 32 22 8B 0000B740 23 23 22 28 B2 37 8C 6A 32 22 BB 23 23 22 0000B750 28 BF FF D9 BA AB AA AB BB AA BA AB BA BA BA 0000B760 AA AB AA BA AA AB AA BA AA AB AA AA AA AB AB AB BB 0000B770 BA AB AB AA BA AB AB AB AB AA AB AA BB A0 + 0000B780 AB AB BA BA AB AA AA BB AA AB A0 +
```

Kemudian kami coba mencatatnya lalu dimasukkan dalam decode chiper identifier, rupanya, ini adalah encrypted text menggunakan Baconian Chiper yang menggunakan kombinasi a dan b setiap 4 huruf. Setelah diedit dan dimasukkan dalam decoder kami pun mendapatkan flagnya.

flag: KashiCTF{THEYWEREREALLLYCOOKING}

3. Corruption



Pada challenge ini kami diberi sebuah file image.iso yang tidak bisa dibuka karena telah corrupted. setelah kami selidiki dan menggunakan berbagai tools, ternyata untuk menyelesaikan soal ini kita hanya perlu untuk menggunakan command hexdump pada terminal linux. berikut adalah commandnya:

Ubuntu `hexdump -C image.iso | less`

Dengan menjalankan command itu kita dapat mendapatkan hex dan diantaranya terdapat flag

```

38 00 2e 00 30 00 |n._.0...6.8...0.|  

64 00 36 00 34 00 |-.1._.a.m.d.6.4.|  

00 00 00 00 00 00 |..d.e.b.....|  

00 00 00 00 00 00 |.....|  

53 43 4b 5f 6d 45 |KashiCTF{FSCK_mE|  

74 5f 42 79 74 33 |_B1T_by_b1t_Byt3|  

00 00 00 00 00 00 |_by_ByT3|.....|  

00 00 00 00 00 00 |.....|  

00 00 08 00 7c 01 |.....|.|

```

flag: KashiCTF{FSCK_mE_B1T_by_b1t_Byt3_by_ByT3}

4. Stego Gambit

Challenge 29 Solves X

Stego Gambit

480

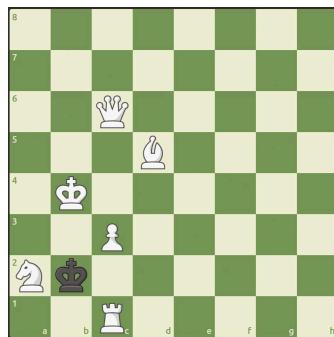
thievius

Do you dare to accept the Stego Gambit? I know you can find the checkmate but the flag!!

 chall.jpg

Flag Submit

Pada challenge ini kami diberi sebuah gambar dari permainan catur seperti pada gambar berikut:



Langkah untuk mendapatkan flag:

1. Analisa Movement skakmat

Bisa kita lihat pada gambar tersebut, langkah agar putih bisa mendapatkan skakmat adalah dengan melakukan move bishop ke h1, lalu king hitam ke a2, dan ditutup oleh queen ke g2 untuk mendapatkan skakmat yang bisa ditulis sebagai berikut:
Bh1Kxa2_Qg2#.

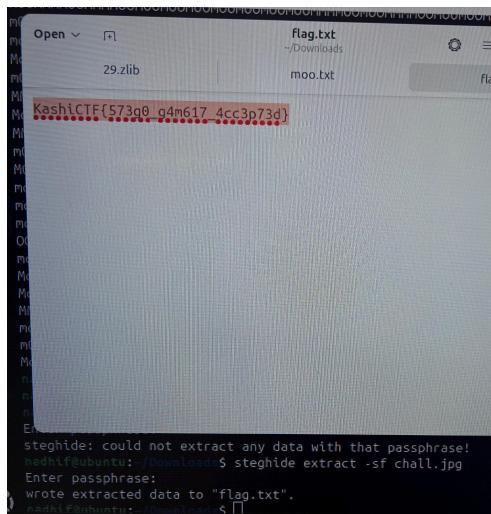
2. Menggunakan steghide

Kami kemudian mencoba untuk melakukan steghide di linux menggunakan password Bh1Kxa2_Qg2# dengan command seperti berikut

Ubuntu steghide extract -sf chall.jpg

Enter passphrase: Bh1Kxa2_Qg2#

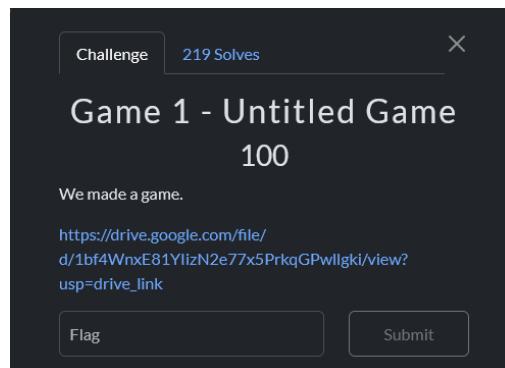
dengan begitu kita akan mendapatkan sebuah flag



flag: KashiCTF{FSCK_mE_B1T_by_b1t_Byt3_by_byT3} *upsolve

Reverse

1. Game 1 - Untitled Game



Pada challenge ini kami diberi sebuah game 3d dan kita harus mencari flagnya dimanapun itu, kita telah mencarinya kedalam gamenya namun kita tidak menemukan apa-apa. setelah kita selidiki ulang menggunakan tools di linux akhirnya kita menemukan bahwa flagnya terdapat pada string dan tidak ada sama sekali didalam gameplaynya. untuk mendapatkan flag kita hanya perlu command grep KashiCTF untuk mencari flag diantara stringnya. berikut adalah commandnya:

Ubuntu **grep -a "KashiCTF" Challgame.exe**

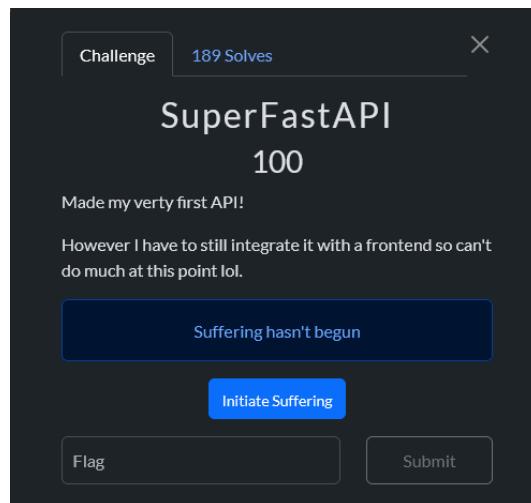
dengan begitu kita akan mendapatkan flagnya

```
nadhif@ubuntu:~/Downloads$ grep -a "KashiCTF" Challgame.exe
var flag = "KashiCTF{N07_1N_7H3_G4M3}" # Get the footstep audio
nadhif@ubuntu:~/Downloads$
```

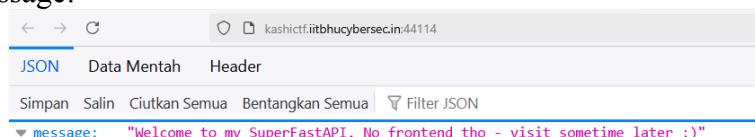
flag: KashiCTF{N07_1N_7H3_G4M3}

Web

1. SuperFastAPI



Pada Challange ini kami diberi sebuah website yang ketika dibuka hanya berisi sebuah message:



setelah bertanya di chatGPT saya disarankan untuk menggunakan endpoint /docs untuk membuka sourcenyanya, dan saya pun dibawa ke page lain

The screenshot shows the SuperFastAPI documentation page. At the top, there's a header with the API name and version (1.0.0 OAS 3.1). Below the header, a note says "Mt first API :)" and "Klik kanan atau tariik turun untuk menampilkan rawat". The main content area is titled "default" and lists several endpoints:

- GET / Root
- GET /get/{username} Get User
- POST /create/{username} Create User
- PUT /update/{username} Update User
- GET /flag/{username} Get Flag

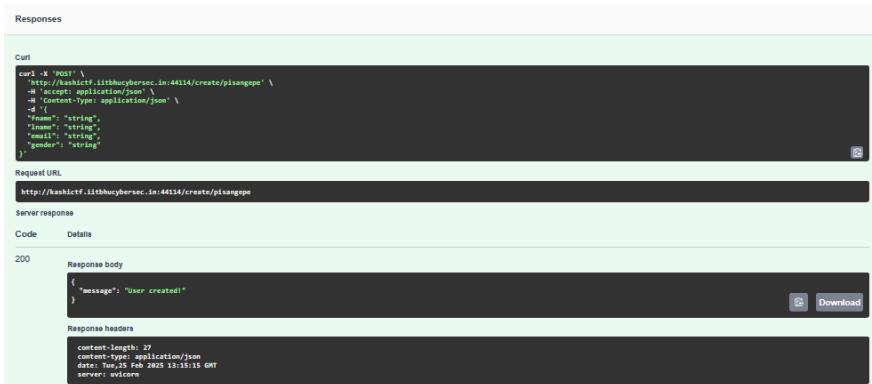
Page ini memiliki beberapa bagian seperti fungsi, setelah mengobservasi dan bertanya-tanya ke AI, sayapun mendapatkan sedikit pencerahan

- GET / Root : untuk mengetahui tampak awal web
- GET /get/{username} : untuk mencari username

- POST /create/{username} : untuk membuat username
- PUT /update/{username} : untuk memperbarui username
- GET /flag/{username} : untuk memperoleh flag dari username

Langkah untuk mendapatkan flag:

1. Create username:



```

Responses

Curl
curl -X 'POST' \
'http://kashictf.lithbucybersec.in:44114/create/pisangepe' \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-d '{
  "name": "string",
  "username": "string",
  "email": "string",
  "gender": "string"
}'

Request URL
http://kashictf.lithbucybersec.in:44114/create/pisangepe

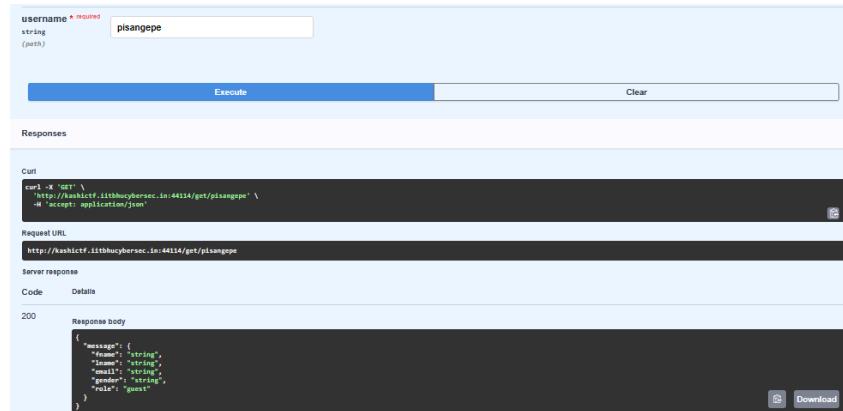
Server response
Code Details
200 Response body
{
  "message": "User created!"
}

Response headers
content-length: 27
content-type: application/json
date: Thu, 25 Feb 2025 13:15:15 GMT
server: uvicorn

```

pertama-tama kita perlu untuk membuat username yang baru di POST /create/{username}

2. Check username



```

Responses

Curl
curl -X 'GET' \
'http://kashictf.lithbucybersec.in:44114/get/pisangepe' \
-H 'Accept: application/json'

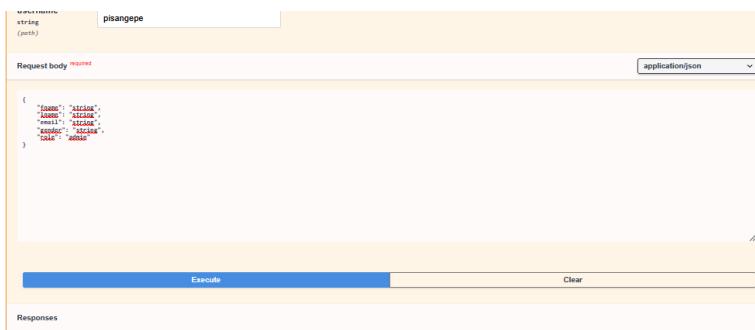
Request URL
http://kashictf.lithbucybersec.in:44114/get/pisangepe

Server response
Code Details
200 Response body
{
  "message": {
    "name": "string",
    "username": "string",
    "email": "string",
    "gender": "string",
    "role": "guest"
  }
}


```

selanjutnya kita perlu mengecek di GET /get/{username}, nah disini kita bisa melihat di bagian response body bahwa kita diberi role sebagai “guest”. sedangkan untuk memperoleh flag di GET /flag/{username} kita harus memiliki role “admin”

3. Update username



```

Responses

String (json)

Request body required
{
  "name": "string",
  "username": "string",
  "email": "string",
  "gender": "string",
  "role": "admin"
}


```

selanjutnya kita perlu mengganti atau menambahkan role kita menjadi “admin” di
PUT /update/{username}

4. Mendapatkan flag

dan langkah terakhir kita hanya perlu menginput username yang telah kita ganti
rolenya di GET /flag/{username} dan kita akan mendapatkan flagnya



The screenshot shows a REST API response interface. At the top, there's a header labeled "Details". Below it, under "Response body", is a JSON object with one key, "message": "KashiCTF{m455_4551gnm3n7_ftw_ZPrTmMBqt}".

```
{  
  "message": "KashiCTF{m455_4551gnm3n7_ftw_ZPrTmMBqt}"  
}
```

flag: KashiCTF{m455_4551gnm3n7_ftw_ZPrTmMBqt}

Osint

1. Who Am I

Pada challenge ini kami diberi sebuah file yang berisikan gambar jalanan dengan latar bangunan di belakangnya. Bangunan tersebut jika dilihat secara teliti bernama Duna House dengan tulisan bazilika@dh di spanduknya.

bazilika@dh

All Images Videos Maps News Shopping Web More ▾ Tools ▾

Tip: Show results in English. You can also learn more about filtering by language.

Duna House
https://dh.hu › Ingatlaniroda :

DH Ingatlaniroda - Bazilika
1065 Budapest 6. kerület Bajcsy-Zsilinszky út 25. Nyitvatartás. Hétfő: 09:00 - 18:00. Kedd: 09:00 - ...

Duna House terletak di Hungaria yang jika dilihat lokasi spesifiknya berada di jalan Bajcsy-Zsilinszky.



Basilika

1065 Budapest distrik ke-6 jalan Bajcsy-Zsilinszky 25.



Karena format flag menggunakan nama lengkap dipisah dengan *_ (underscore)*, maka ditemukan bahwa flagnya adalah:

These are results for **Bajcsy_Zsilinszky**
Search instead for Bajcsy_Zsilinszky

Endre Bajcsy-Zsilinszky
Hungarian Politician :

flag: KashiCTF{Endre_Bajcsy_Zsilinszky}

2. Old Diner

Challenge 56 Solves X

Old Diner
411
shinigami-777

My friend once visited this place that served ice cream with coke. He said he had the best Greek omelette of his life and called it a very american experience. Can you find the name of the diner and the amount he paid?

Flag Format: KashiCTF{Name_of_Diner_Amount}

For clarification on the flag format The diner's name is in title case with spaces replaced by underscores. The amount is without currency sign, and in decimal, correct to two decimal places, i.e.
KashiCTF{Full_Diner_Name_xx.xx}

Flag Submit

Pada challenge ini kami diberi sebuah informasi bahwa ada sebuah tempat yang menyediakan ice cream with coke dan memiliki menu Greek Omelete. Setelah meminta bantuan ChatGPT dengan berdasarkan informasi yang kami dapatkan, kami menemukan sebuah tempat bernama Lexington Candy Shop.

Old Diner

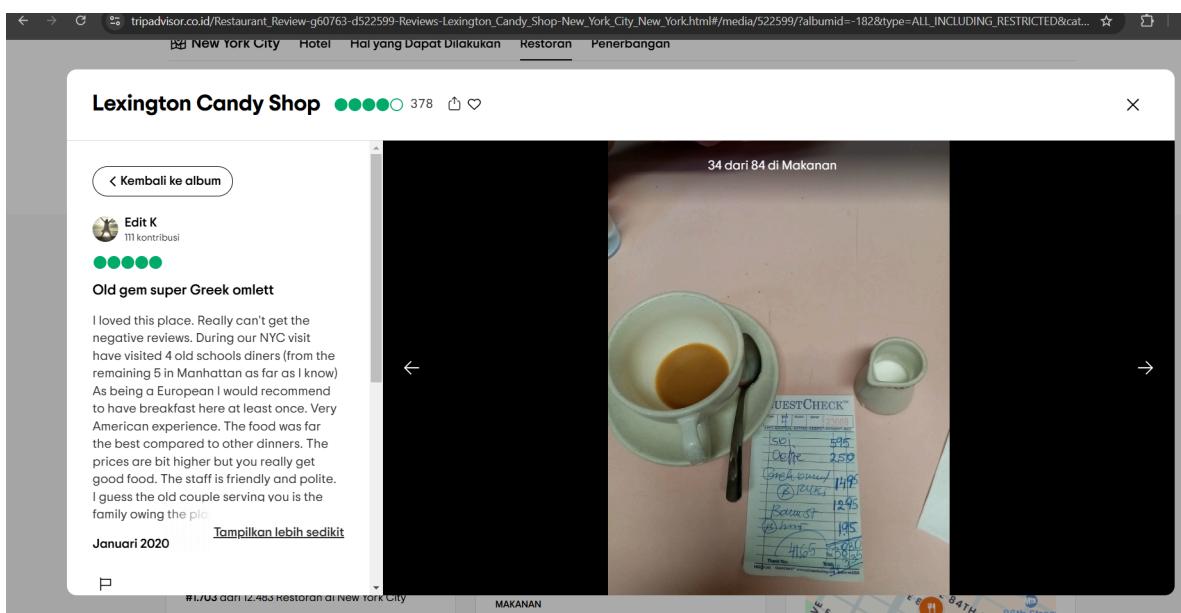
It sounds like your friend visited the Lexington Candy Shop, a historic diner on the Upper East Side of New York City. Established in 1925, this nearly century-old establishment is renowned for serving Coca-Cola the old-fashioned way, mixing syrup with seltzer water, and offering classic American dishes, including a variety of omelets. The combination of traditional soda fountain drinks and hearty meals provides a quintessential American dining experience.

Another possibility is the Bridgehampton Candy Kitchen, an old-school Greek diner and ice cream parlor in Bridgehampton, New York. Known for its amazing breakfasts, including delicious omelettes, and a variety of ice cream treats, this establishment offers a blend of Greek and American culinary traditions.

If your friend recalls more specific details, such as the exact location or unique features of the diner, I can help pinpoint the exact establishment.

Sumber

Lalu, untuk mengetahui total yang dibayar kami browsing menggunakan google lalu sampai pada bagian ulasan dari Lexington Candy Shop. Setelah meneliti dengan seksama kami pun mendapatkan total yang dibayar, sesuai dengan menuanya.



flag: KashiCTF{Lexington_Candy_Shop_41.65} *upsolve