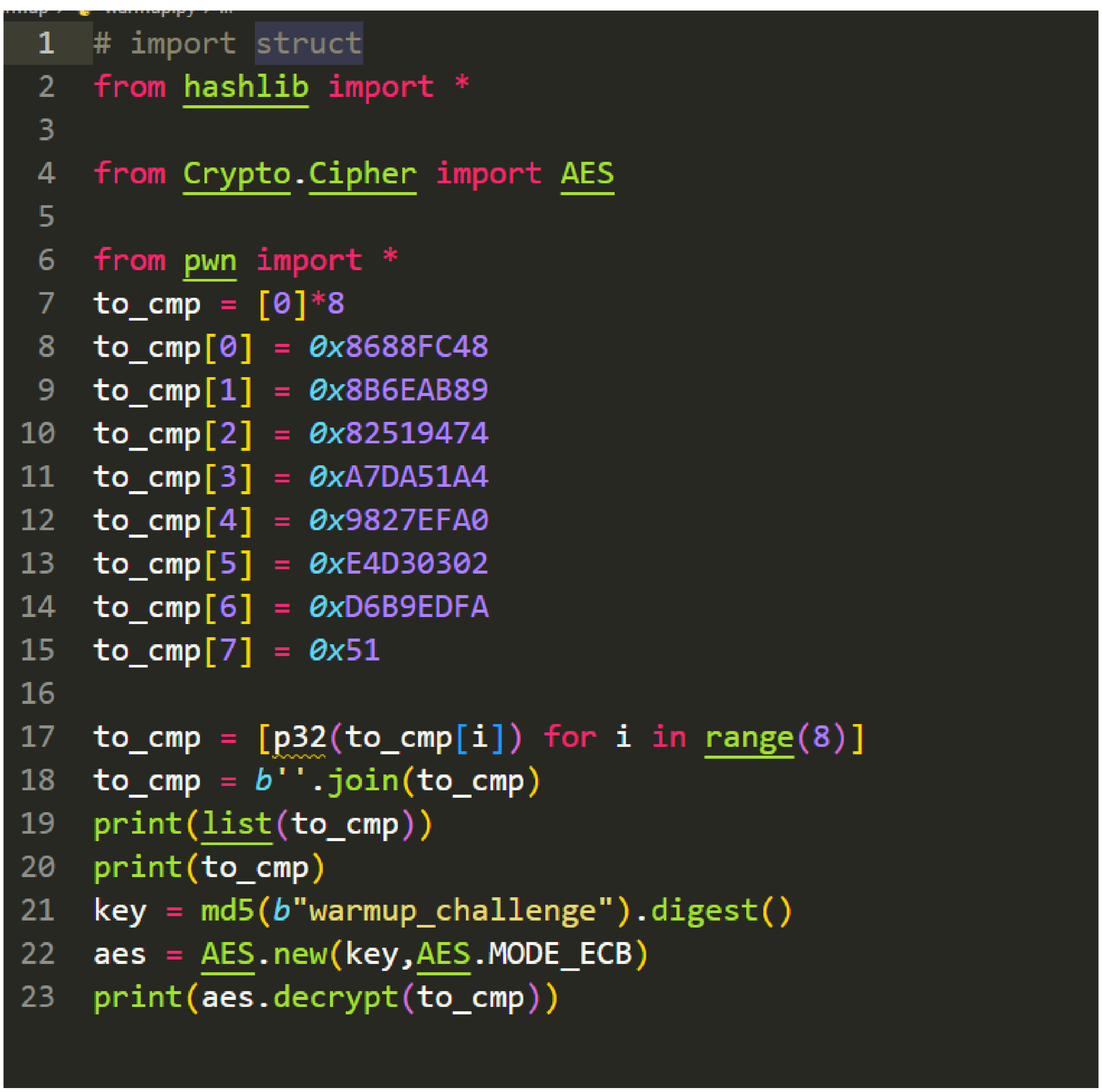
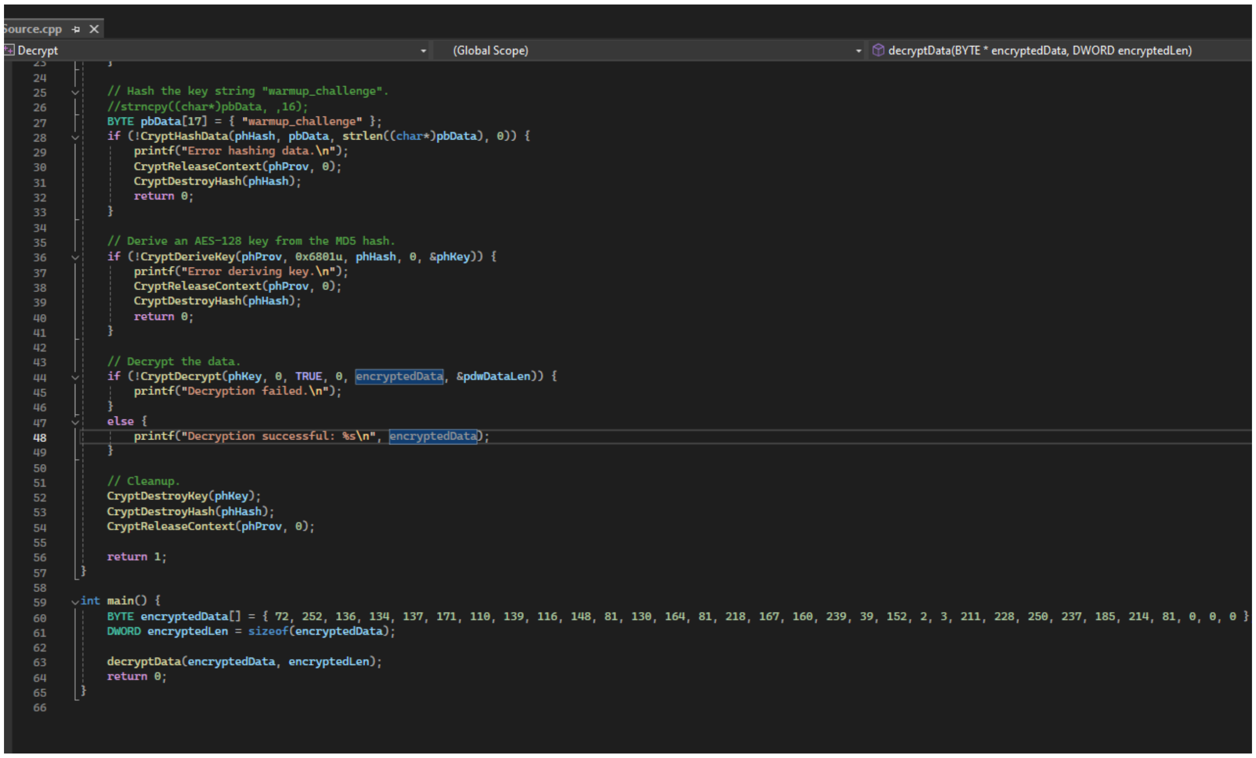
Warmup:

Câu thuật toán có sẵn trong hàm main nên mình implement lại trong python, tuy nhiên lại ko ra.



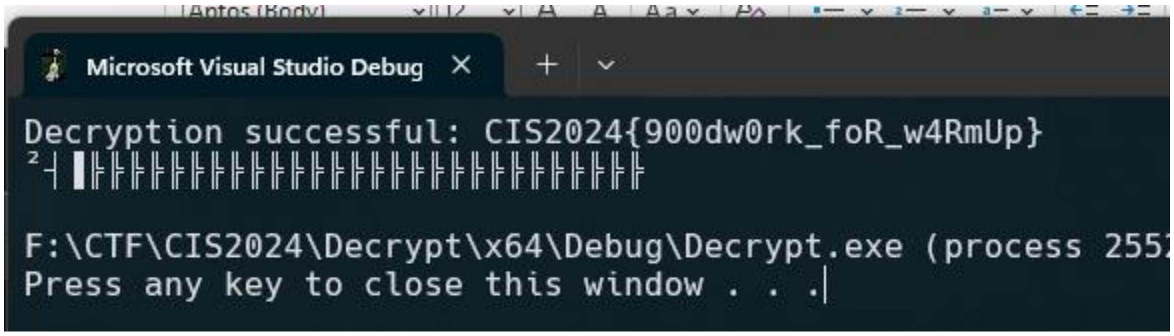
```python
# import struct
from hashlib import *

from Crypto.Cipher import AES

from pwn import *
to_cmp = [0]*8
to_cmp[0] = 0x8688FC48
to_cmp[1] = 0x8B6EAB89
to_cmp[2] = 0x82519474
to_cmp[3] = 0xA7DA51A4
to_cmp[4] = 0x9827EFA0
to_cmp[5] = 0xE4D30302
to_cmp[6] = 0xD6B9EDFA
to_cmp[7] = 0x51

to_cmp = [p32(to_cmp[i]) for i in range(8)]
to_cmp = b''.join(to_cmp)
print(list(to_cmp))
print(to_cmp)
key = md5(b"warmup_challenge").digest()
aes = AES.new(key,AES.MODE_ECB)
print(aes.decrypt(to_cmp))
```

Tuy nhiên mình biết lý do, Do đó mình đã implement lại trong C++, dùng API của windows và được flag.



```
    }

    // Hash the key string "warmup_challenge".
    //strncpy((char*)pbData, ,16);
    BYTE pbData[17] = { "warmup_challenge" };
    if (!CryptHashData(phHash, pbData, strlen((char*)pbData), 0)) {
        printf("Error hashing data.\n");
        CryptReleaseContext(phProv, 0);
        CryptDestroyHash(phHash);
        return 0;
    }

    // Derive an AES-128 key from the MD5 hash.
    if (!CryptDeriveKey(phProv, 0x6801u, phHash, 0, &phKey)) {
        printf("Error deriving key.\n");
        CryptReleaseContext(phProv, 0);
        CryptDestroyHash(phHash);
        return 0;
    }

    // Decrypt the data.
    if (!CryptDecrypt(phKey, 0, TRUE, 0, encryptedData, &pdwDataLen)) {
        printf("Decryption failed.\n");
    }
    else {
        printf("Decryption successful: %s\n", encryptedData);
    }

    // Cleanup.
    CryptDestroyKey(phKey);
    CryptDestroyHash(phHash);
    CryptReleaseContext(phProv, 0);

    return 1;
}

int main() {
    BYTE encryptedData[] = { 72, 252, 136, 134, 137, 171, 110, 139, 116, 148, 81, 130, 164, 81, 218, 167, 160, 239, 39, 152, 2, 3, 211, 228, 250, 237, 185, 214, 81, 0, 0, 0 };
    DWORD encryptedLen = sizeof(encryptedData);

    decryptData(encryptedData, encryptedLen);
    return 0;
}
```

Source.cpp
Decrypt
(Global Scope)
decryptData(BYTE * encryptedData, DWORD encryptedLen)



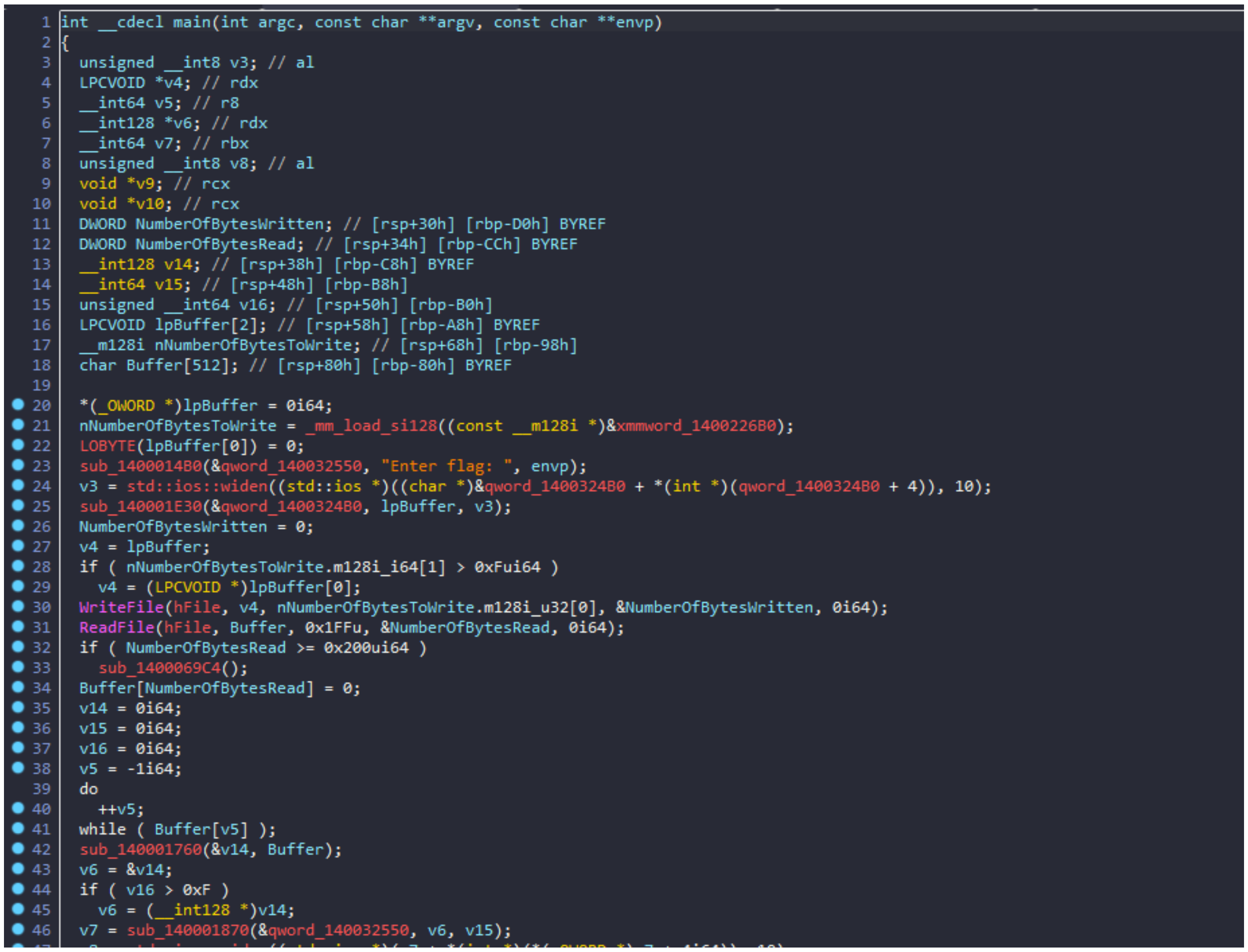
Microsoft Visual Studio Debug

```
Decryption successful: CIS2024{900dw0rk_foR_w4RmUp}
²┤▐╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠╠

F:\CTF\CIS2024\Decrypt\x64\Debug\Decrypt.exe (process 255
Press any key to close this window . . .
```

Flag: CIS2024{900dw0rk_foR_w4RmUp}

Nát đờ:

Câu này thuật toán có đôi chút phức tạp hơn:



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned __int8 v3; // al
  LPCVOID *v4; // rdx
  __int64 v5; // r8
  __int128 *v6; // rdx
  __int64 v7; // rbx
  unsigned __int8 v8; // al
  void *v9; // rcx
  void *v10; // rcx
  DWORD NumberOfBytesWritten; // [rsp+30h] [rbp-D0h] BYREF
  DWORD NumberOfBytesRead; // [rsp+34h] [rbp-CCh] BYREF
  __int128 v14; // [rsp+38h] [rbp-C8h] BYREF
  __int64 v15; // [rsp+48h] [rbp-B8h]
  unsigned __int64 v16; // [rsp+50h] [rbp-B0h]
  LPCVOID lpBuffer[2]; // [rsp+58h] [rbp-A8h] BYREF
  __m128i nNumberOfBytesToWrite; // [rsp+68h] [rbp-98h]
  char Buffer[512]; // [rsp+80h] [rbp-80h] BYREF

  *(_OWORD *)lpBuffer = 0i64;
  nNumberOfBytesToWrite = _mm_load_si128((const __m128i *)&xmmword_1400226B0);
  LOBYTE(lpBuffer[0]) = 0;
  sub_1400014B0(&qword_140032550, "Enter flag: ", envp);
  v3 = std::ios::widen((std::ios *)((char *)&qword_1400324B0 + *(int *)(qword_1400324B0 + 4)), 10);
  sub_140001E30(&qword_1400324B0, lpBuffer, v3);
  NumberOfBytesWritten = 0;
  v4 = lpBuffer;
  if ( nNumberOfBytesToWrite.m128i_i64[1] > 0xFui64 )
    v4 = (LPCVOID *)lpBuffer[0];
  WriteFile(hFile, v4, nNumberOfBytesToWrite.m128i_u32[0], &NumberOfBytesWritten, 0i64);
  ReadFile(hFile, Buffer, 0x1FFu, &NumberOfBytesRead, 0i64);
  if ( NumberOfBytesRead >= 0x200ui64 )
    sub_1400069C4();
  Buffer[NumberOfBytesRead] = 0;
  v14 = 0i64;
  v15 = 0i64;
  v16 = 0i64;
  v5 = -1i64;
  do
    ++v5;
  while ( Buffer[v5] );
  sub_140001760(&v14, Buffer);
  v6 = &v14;
  if ( v16 > 0xF )
    v6 = (__int128 *)v14;
  v7 = sub_140001870(&qword_140032550, v6, v15);
```

Câu này load library dynamic nên mình chỉ cần tìm xung quanh những hàm khả nghi rồi đặt breakpoint:

Trong đó mình thấy bài này có dùng bcrypt và một số hàm khác.

BCryptOpenAlgorithmProvider

Idea câu này là dùng pipe để giao tiếp và check flag, trong đó một hàm khác sẽ handle việc này.

Trong lúc debug mình phát hiện toán sơ bộ của nó:

#include <windows.h>

#include <bcrypt.h>

#include <stdio.h>

#include <string.h>

```
#pragma comment(lib, "bcrypt.lib")

#define NT_SUCCESS(Status) (((NTSTATUS)(Status)) >= 0)

#define AES_KEY_SIZE 16  // AES-128 key
#define SALT_SIZE 22     // Length of the salt in bytes
#define IV_SIZE 16       // AES block size for IV
#define BUFFER_SIZE 64   // Example buffer size to encrypt

void handleError(const char* errorMessage, NTSTATUS status) {
  printf("%s failed with NTSTATUS code: 0x%x\n", errorMessage, status);
  exit(EXIT_FAILURE);
}

void printBuffer(const char* title, const BYTE* buffer, DWORD len) {
  printf("%s: ", title);
  for (DWORD i = 0; i < len; i++) {
    printf("%02x ", buffer[i]);
  }
  printf("\n");
}

int main() {
  NTSTATUS status;
  BCRYPT_ALG_HANDLE hAesAlg = NULL;
```

```
    BCRYPT_HASH_HANDLE hKeyDerivationAlg = NULL;

    BCRYPT_KEY_HANDLE hKey = NULL;

    BYTE pbKey[AES_KEY_SIZE] = { 0 };

    BYTE pbIV[IV_SIZE] = { 0 }; // Initialization Vector (IV)

    BYTE pbSalt[SALT_SIZE] = "the co lam duoc khong";

    BYTE pbPassword[] = "tin chuan chua";

    DWORD cbDerivedKey = AES_KEY_SIZE;

    BYTE buffer[BUFFER_SIZE] = "This is the data to encrypt!";

    DWORD cbBuffer = (DWORD)strlen((char*)buffer) + 1;

    DWORD cbCipherText = 0;

    BYTE encryptedBuffer[BUFFER_SIZE] = { 0 };


    // Open an AES algorithm handle

    status = BCryptOpenAlgorithmProvider(&hAesAlg, BCRYPT_AES_ALGORITHM, NULL, 0);

    if (!NT_SUCCESS(status)) handleError("BCryptOpenAlgorithmProvider", status);


    // Open an PBKDF2 key derivation algorithm handle

    status = BCryptOpenAlgorithmProvider(&hKeyDerivationAlg,
BCRYPT_PBKDF2_ALGORITHM, NULL, 0);

    if (!NT_SUCCESS(status)) handleError("BCryptOpenAlgorithmProvider (PBKDF2)", status);


    // Derive the AES key using PBKDF2 (BCryptDeriveKeyPBKDF2)

    status = BCryptDeriveKeyPBKDF2(

        hKeyDerivationAlg,

        pbPassword, (ULONG)strlen((char*)pbPassword),   // Password

        pbSalt, SALT_SIZE,                       // Salt
        10000,                                  // Iterations (work factor)
        pbKey, cbDerivedKey,                    // Output buffer for derived key
        0);                                     // Flags
    if (!NT_SUCCESS(status)) handleError("BCryptDeriveKeyPBKDF2", status);

    printBuffer("Derived Key", pbKey, cbDerivedKey);

    // Generate a random IV (Initialization Vector)
    status = BCryptGenRandom(NULL, pbIV, IV_SIZE,
BCRYPT_USE_SYSTEM_PREFERRED_RNG);
    if (!NT_SUCCESS(status)) handleError("BCryptGenRandom (IV)", status);

    printBuffer("Initialization Vector", pbIV, IV_SIZE);

    // Generate an AES key from the derived key
    status = BCryptGenerateSymmetricKey(hAesAlg, &hKey, NULL, 0, pbKey, cbDerivedKey,
0);
    if (!NT_SUCCESS(status)) handleError("BCryptGenerateSymmetricKey", status);

    // Encrypt the buffer
    status = BCryptEncrypt(
        hKey,                   // Handle to the encryption key
        buffer, cbBuffer,       // Plaintext to encrypt
        NULL,                   // Padding info (none in CBC mode)
        pbIV, IV_SIZE,          // Initialization vector
        encryptedBuffer, BUFFER_SIZE,  // Output buffer for ciphertext
        &cbCipherText,          // Ciphertext size
        BCRYPT_BLOCK_PADDING); // Padding flag for block size
    if (!NT_SUCCESS(status)) handleError("BCryptEncrypt", status);

    printBuffer("Encrypted Data", encryptedBuffer, cbCipherText);

    // Cleanup
    if (hKey) BCryptDestroyKey(hKey);
    if (hAesAlg) BCryptCloseAlgorithmProvider(hAesAlg, 0);
    if (hKeyDerivationAlg) BCryptCloseAlgorithmProvider(hKeyDerivationAlg, 0);

    return 0;
}
```

Mình có thể có nhiều cách làm, tuy nhiên cách của mình đơn giản chỉ là debug lại trước hàm Encrypt, sau đó lấy key và IV.

Và dùng script này để decrypt:

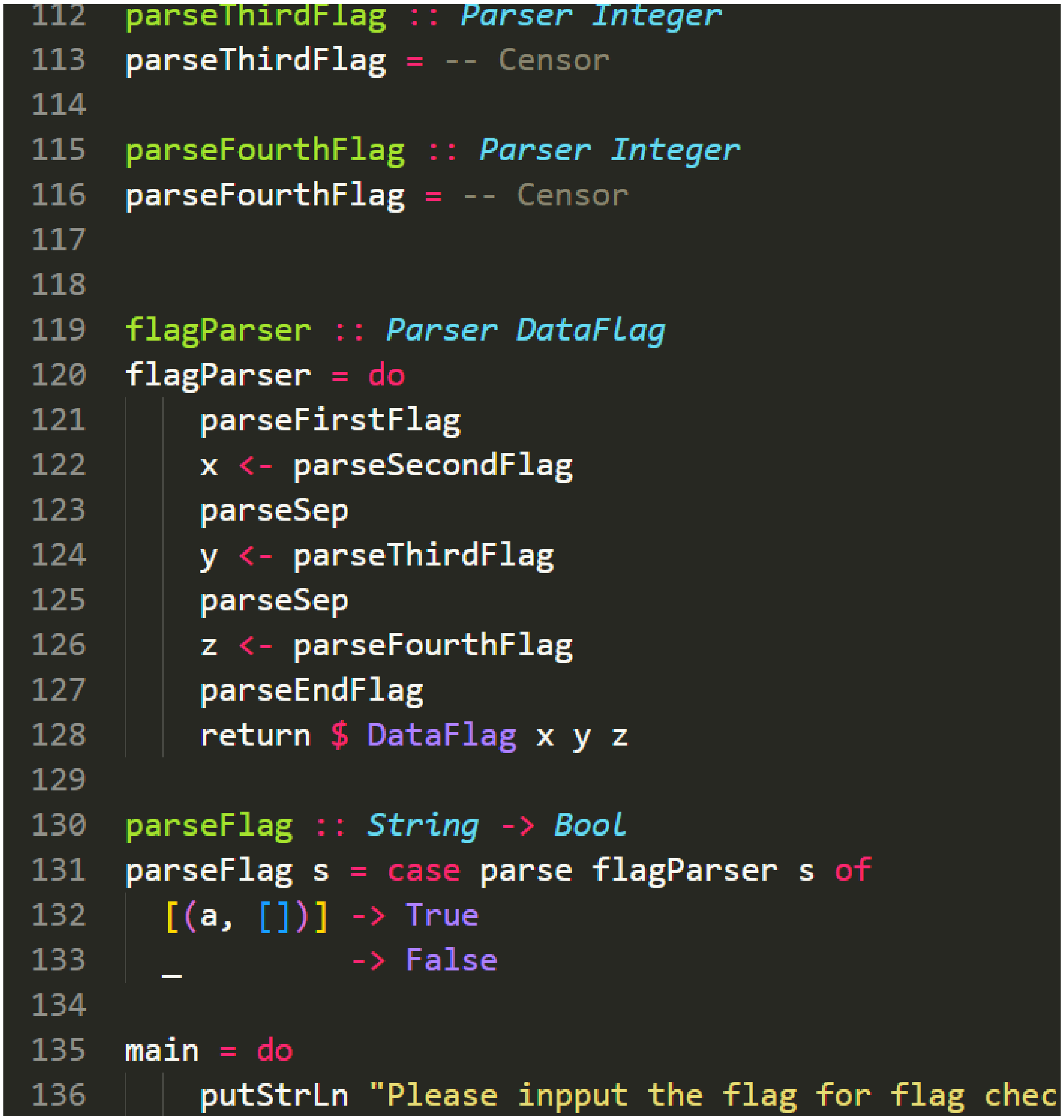
```
from hashlib import sha512
from pwn import p32
t0 = b'tin chuan chua'
t1 = b'the co lam duoc khong'
hashed = sha512(t0).digest()

print(hashed,len(hashed))
print(len(t0), len(t1))

to_cmp = [
    0x441C50DD,
    0x55783DC1,

    0x36EB684B,

    0x287176F,

    0x5A411788,

    0x5135A959,

    0x6197354E,

    0xEBF5B04B]

to_cmp = [p32(i) for i in to_cmp]

to_cmp = b''.join(to_cmp)

print(to_cmp)

# print(list(to_cmp))


from Crypto.Cipher import AES


buf =
b'B\xf9>\xdd\xa2\\\x08\xae\xe738\xee\x00\xab\xbe\n\xe7\xb1\x8c\xba\x1a\x05\xe3N\x98`\xff\x87\x7f\x10\x9c\xaa\xaa\xe2\\\x1a\r\xf8j\xa0\xc6]5\xc4\xc9\x80\x0e\x17'


key = buf[:32]

iv = buf[32:]

aes = AES.new(key,AES.MODE_CBC,iv)


print(aes.decrypt(to_cmp))
```

Parser:

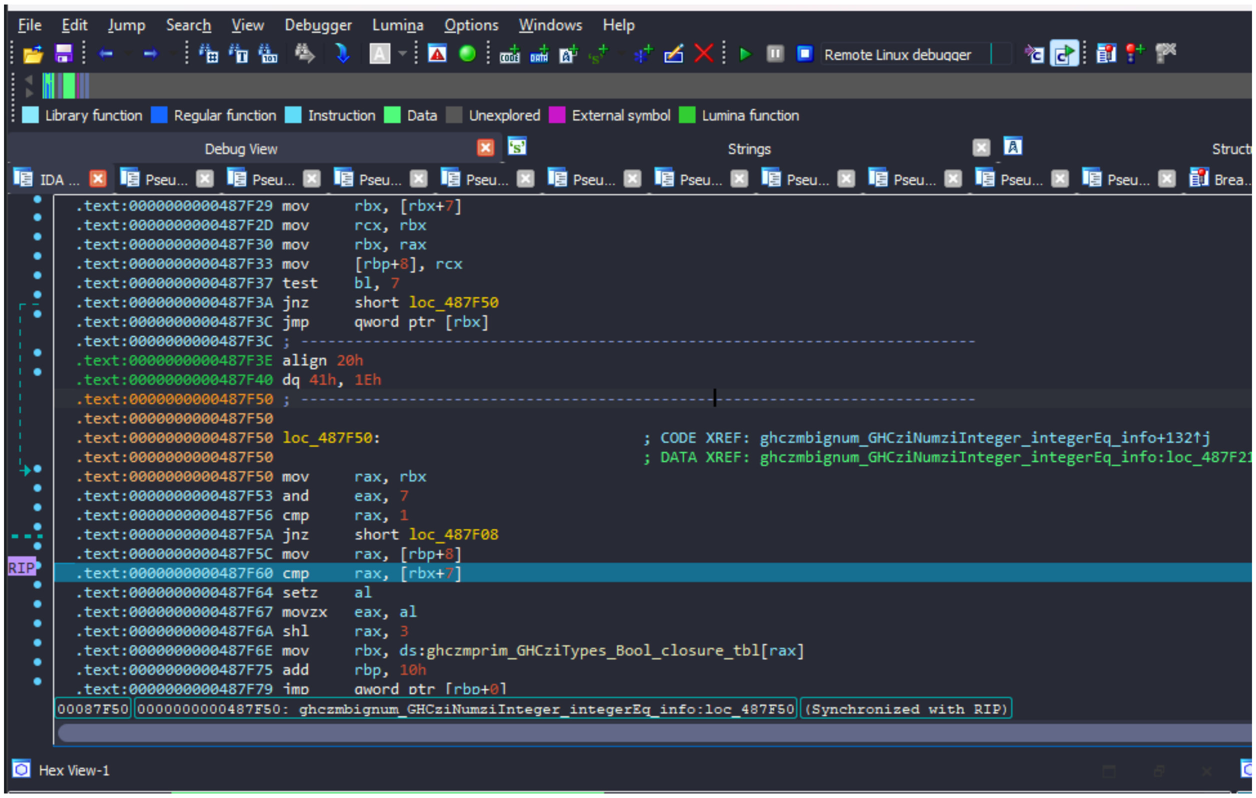
Idea câu này là mình dùng IDA revese thuần, trong đó flag chia làm 3 part

Dựa theo 3 part trong file .hs, trace trong IDA, mình sẽ đặt hardware breakpoint ở từng part

Sau đó sẽ trace từng instruction.

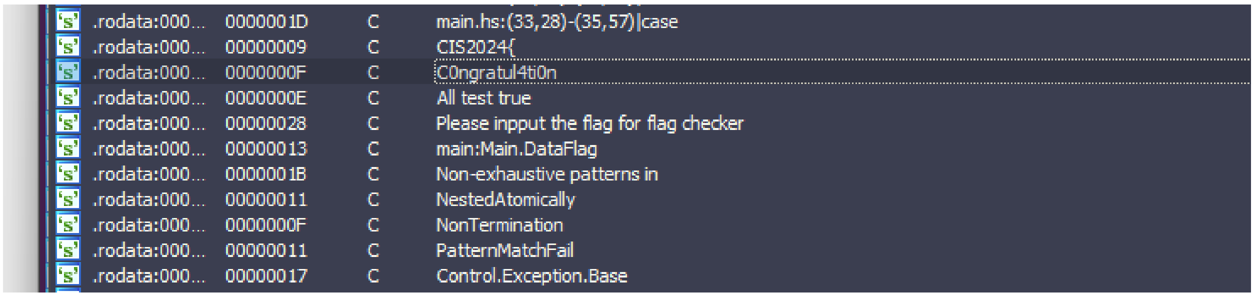


```haskell
parseThirdFlag :: Parser Integer
parseThirdFlag = -- Censor

parseFourthFlag :: Parser Integer
parseFourthFlag = -- Censor


flagParser :: Parser DataFlag
flagParser = do
    parseFirstFlag
    x <- parseSecondFlag
    parseSep
    y <- parseThirdFlag
    parseSep
    z <- parseFourthFlag
    parseEndFlag
    return $ DataFlag x y z

parseFlag :: String -> Bool
parseFlag s = case parse flagParser s of
  [(a, [])] -> True
  _         -> False

main = do
    putStrLn "Please inpput the flag for flag chec
```



```
.text:0000000000487F29 mov     rbx, [rbx+7]
.text:0000000000487F2D mov     rcx, rbx
.text:0000000000487F30 mov     rbx, rax
.text:0000000000487F33 mov     [rbp+8], rcx
.text:0000000000487F37 test    bl, 7
.text:0000000000487F3A jnz     short loc_487F50
.text:0000000000487F3C jmp     qword ptr [rbx]
.text:0000000000487F3C ; ---------------------------------------------------------------------------
.text:0000000000487F3E align 20h
.text:0000000000487F40 dq 41h, 1Eh
.text:0000000000487F50 ; ---------------------------------------------------------------------------
.text:0000000000487F50
.text:0000000000487F50 loc_487F50:                             ; CODE XREF: ghczmbignum_GHCziNumziInteger_integerEq_info+132↑j
.text:0000000000487F50                                         ; DATA XREF: ghczmbignum_GHCziNumziInteger_integerEq_info:loc_487F2
.text:0000000000487F50 mov     rax, rbx
.text:0000000000487F53 and     eax, 7
.text:0000000000487F56 cmp     rax, 1
.text:0000000000487F5A jnz     short loc_487F08
.text:0000000000487F5C mov     rax, [rbp+8]
.text:0000000000487F60 cmp     rax, [rbx+7]
.text:0000000000487F64 setz    al
.text:0000000000487F67 movzx   eax, al
.text:0000000000487F6A shl     rax, 3
.text:0000000000487F6E mov     rbx, ds:ghczmprim_GHCziTypes_Bool_closure_tbl[rax]
.text:0000000000487F75 add     rbp, 10h
.text:0000000000487F79 jmp     qword ptr [rbp+0]
00087F50 0000000000487F50: ghczmbignum_GHCziNumziInteger_integerEq_info:loc_487F50 (Synchronized with RIP)
```

Tìm string mình được part đầu tiên:



```
.rodata:000...   0000001D   C   main.hs:(33,28)-(35,57)|case
.rodata:000...   00000009   C   CIS2024{
.rodata:000...   0000000F   C   C0ngratul4ti0n
.rodata:000...   0000000E   C   All test true
.rodata:000...   00000028   C   Please inpput the flag for flag checker
.rodata:000...   00000013   C   main:Main.DataFlag
.rodata:000...   0000001B   C   Non-exhaustive patterns in
.rodata:000...   00000011   C   NestedAtomically
.rodata:000...   0000000F   C   NonTermination
.rodata:000...   00000011   C   PatternMatchFail
.rodata:000...   00000017   C   Control.Exception.Base
```

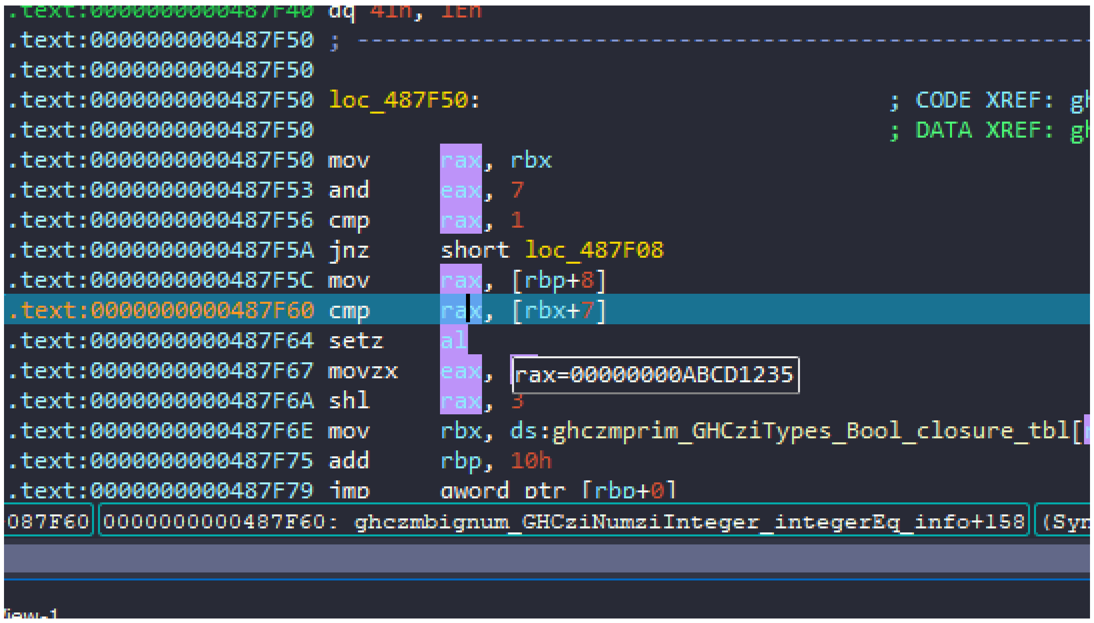
Ở part thứ 2, sau khi mình debug thì mình biết part đó chỉ nhận toàn số, sau đó convert sang decimal và compare với 1 số cho trước: 0x1337c0de

Từ đây chỉ cần đổi số này sang decimal và được part 2:

CIS2024{C0ngratul4ti0n_322420958

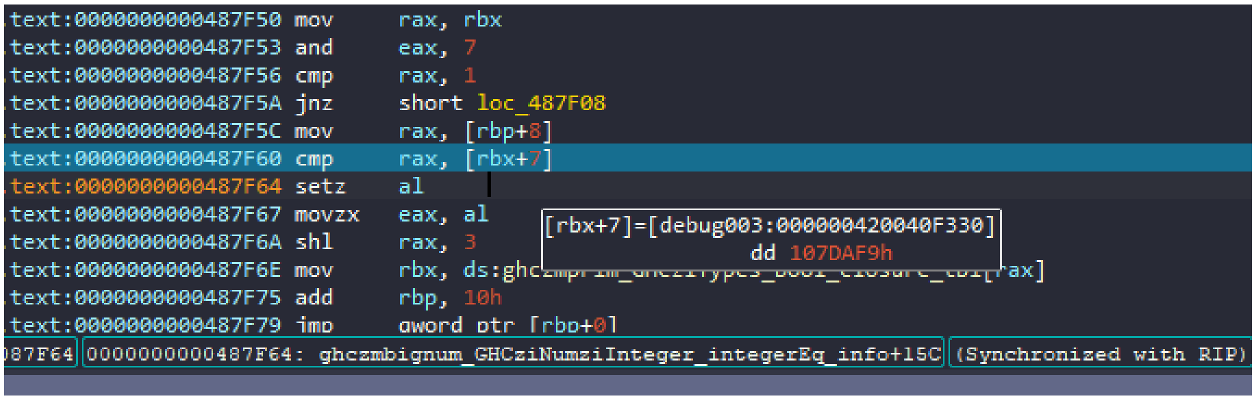
Còn part cuối, dựa theo idea 2 part trên, tuy nhiên phần này có hơi tricky xíu là nó sẽ đổi từ hexadecimal sang interger, sau đó cộng them một

Mình trace thì tìm được số này (nhập abcd1234)



```
.text:0000000000487F40 dq 41h, 1Eh
.text:0000000000487F50 ; ----------------------------------------------------
.text:0000000000487F50
.text:0000000000487F50 loc_487F50:                           ; CODE XREF: gh
.text:0000000000487F50                                       ; DATA XREF: gh
.text:0000000000487F50 mov     rax, rbx
.text:0000000000487F53 and     eax, 7
.text:0000000000487F56 cmp     rax, 1
.text:0000000000487F5A jnz     short loc_487F08
.text:0000000000487F5C mov     rax, [rbp+8]
.text:0000000000487F60 cmp     rax, [rbx+7]
.text:0000000000487F64 setz    al
.text:0000000000487F67 movzx   eax, rax=00000000ABCD1235
.text:0000000000487F6A shl     rax, 3
.text:0000000000487F6E mov     rbx, ds:ghczmprim_GHCziTypes_Bool_closure_tbl[
.text:0000000000487F75 add     rbp, 10h
.text:0000000000487F79 jmp     qword ptr [rbp+0]
087F60 0000000000487F60: ghczmbignum_GHCziNumziInteger_integerEq_info+158 (Syn
```

Sau đó chỉ cần lấy số đã compare, trừ đi 1, ghép được part cuối:

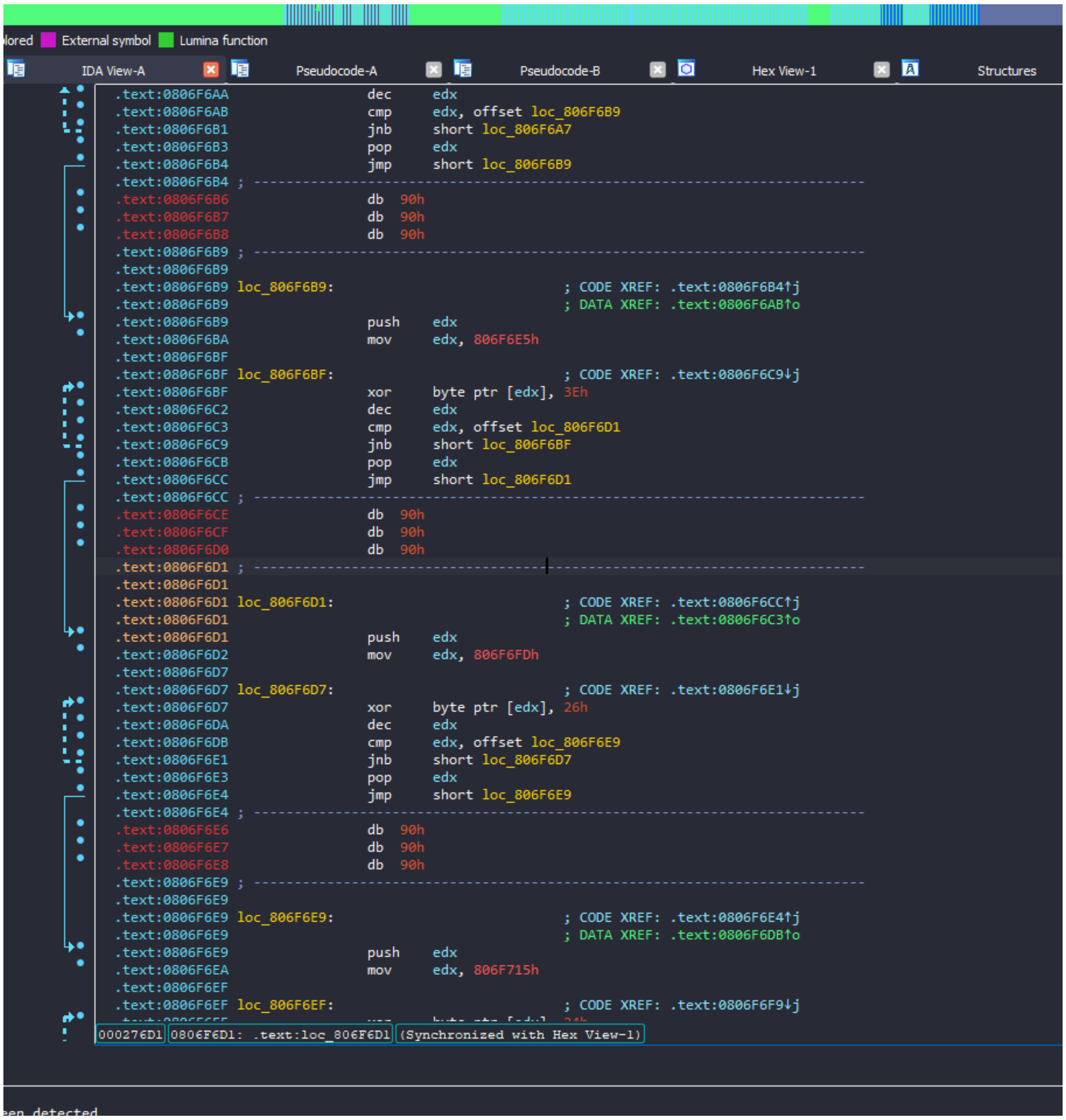


```
text:0000000000487F50 mov     rax, rbx
text:0000000000487F53 and     eax, 7
text:0000000000487F56 cmp     rax, 1
text:0000000000487F5A jnz     short loc_487F08
text:0000000000487F5C mov     rax, [rbp+8]
text:0000000000487F60 cmp     rax, [rbx+7]
text:0000000000487F64 setz    al
text:0000000000487F67 movzx   eax, al   [rbx+7]=[debug003:000000420040F330]
text:0000000000487F6A shl     rax, 3              dd 107DAF9h
text:0000000000487F6E mov     rbx, ds:ghczmprim_GHCziTypes_Bool_closure_tbl[rax]
text:0000000000487F75 add     rbp, 10h
text:0000000000487F79 jmp     qword ptr [rbp+0]
87F64 0000000000487F64: ghczmbignum_GHCziNumziInteger_integerEq_info+15C (Synchronized with RIP)
```

Flag: CIS2024{C0ngratul4ti0n_322420958_0107daf8}

Flatten:

Câu này thì thuật toán cực kì đơn giản, đây là code sau khi mình flatten:



```
.text:0806F6AA                 dec     edx
.text:0806F6AB                 cmp     edx, offset loc_806F6B9
.text:0806F6B1                 jnb     short loc_806F6A7
.text:0806F6B3                 pop     edx
.text:0806F6B4                 jmp     short loc_806F6B9
.text:0806F6B4 ; ---------------------------------------------------------------------------
.text:0806F6B6                 db  90h
.text:0806F6B7                 db  90h
.text:0806F6B8                 db  90h
.text:0806F6B9 ; ---------------------------------------------------------------------------
.text:0806F6B9
.text:0806F6B9 loc_806F6B9:                            ; CODE XREF: .text:0806F6B4↑j
.text:0806F6B9                                         ; DATA XREF: .text:0806F6AB↑o
.text:0806F6B9                 push    edx
.text:0806F6BA                 mov     edx, 806F6E5h
.text:0806F6BF
.text:0806F6BF loc_806F6BF:                            ; CODE XREF: .text:0806F6C9↓j
.text:0806F6BF                 xor     byte ptr [edx], 3Eh
.text:0806F6C2                 dec     edx
.text:0806F6C3                 cmp     edx, offset loc_806F6D1
.text:0806F6C9                 jnb     short loc_806F6BF
.text:0806F6CB                 pop     edx
.text:0806F6CC                 jmp     short loc_806F6D1
.text:0806F6CC ; ---------------------------------------------------------------------------
.text:0806F6CE                 db  90h
.text:0806F6CF                 db  90h
.text:0806F6D0                 db  90h
.text:0806F6D1 ; ---------------------------------------------------------------------------
.text:0806F6D1
.text:0806F6D1 loc_806F6D1:                            ; CODE XREF: .text:0806F6CC↑j
.text:0806F6D1                                         ; DATA XREF: .text:0806F6C3↑o
.text:0806F6D1                 push    edx
.text:0806F6D2                 mov     edx, 806F6FDh
.text:0806F6D7
.text:0806F6D7 loc_806F6D7:                            ; CODE XREF: .text:0806F6E1↓j
.text:0806F6D7                 xor     byte ptr [edx], 26h
.text:0806F6DA                 dec     edx
.text:0806F6DB                 cmp     edx, offset loc_806F6E9
.text:0806F6E1                 jnb     short loc_806F6D7
.text:0806F6E3                 pop     edx
.text:0806F6E4                 jmp     short loc_806F6E9
.text:0806F6E4 ; ---------------------------------------------------------------------------
.text:0806F6E6                 db  90h
.text:0806F6E7                 db  90h
.text:0806F6E8                 db  90h
.text:0806F6E9 ; ---------------------------------------------------------------------------
.text:0806F6E9
.text:0806F6E9 loc_806F6E9:                            ; CODE XREF: .text:0806F6E4↑j
.text:0806F6E9                                         ; DATA XREF: .text:0806F6DB↑o
.text:0806F6E9                 push    edx
.text:0806F6EA                 mov     edx, 806F715h
.text:0806F6EF
.text:0806F6EF loc_806F6EF:                            ; CODE XREF: .text:0806F6F9↓j
000276D1 0806F6D1: .text:loc_806F6D1 (Synchronized with Hex View-1)
```

```
een detected
```

Thuật toán và Script flatten:

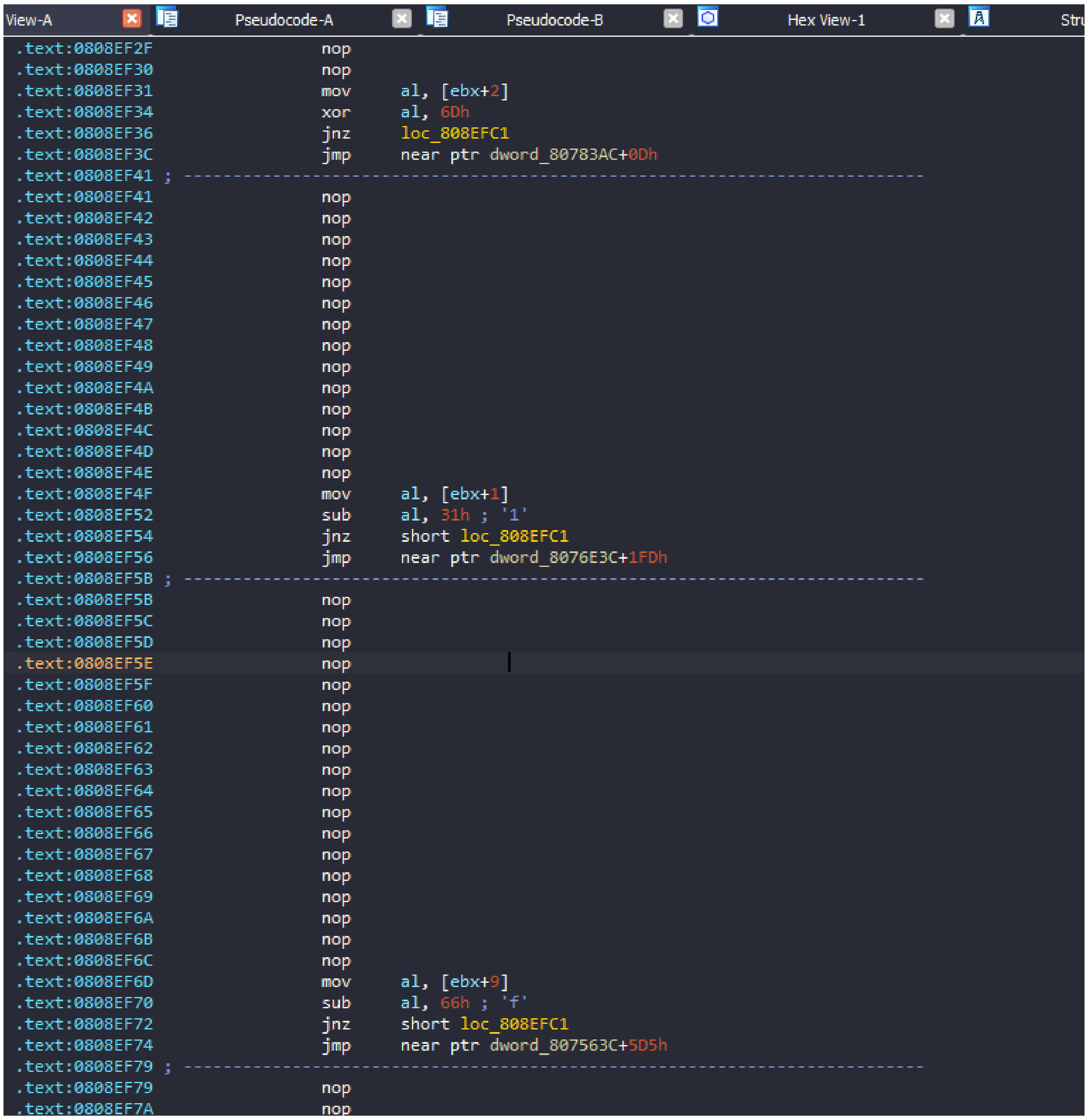
from pwn import *

# chunk_addr = 0x806D961

chunk_addr = *0x*808EFC1

```
def patch__(edx, end, x):
    while True:
        patch_byte(edx, ord(get_bytes(edx, 1)) ^ x)
        edx-=1
        if (edx == end-1):
            break

for i in range(6301):
    start = u32(get_bytes(chunk_addr + 0x2, 4))
    end = u32(get_bytes(chunk_addr + 0xc, 4))
    x = u8(get_bytes(chunk_addr + 0x8, 1))
    print(i,hex(chunk_addr))
    patch__(start, end, x)
    chunk_addr += 0x18
```

Giữa 2 chunk address 0x806D961 và *0x*808EFC1 có đoạn check flag:



```
.text:0808EF2F                 nop
.text:0808EF30                 nop
.text:0808EF31                 mov     al, [ebx+2]
.text:0808EF34                 xor     al, 6Dh
.text:0808EF36                 jnz     loc_808EFC1
.text:0808EF3C                 jmp     near ptr dword_80783AC+0Dh
.text:0808EF41 ; ---------------------------------------------------------------------------
.text:0808EF41                 nop
.text:0808EF42                 nop
.text:0808EF43                 nop
.text:0808EF44                 nop
.text:0808EF45                 nop
.text:0808EF46                 nop
.text:0808EF47                 nop
.text:0808EF48                 nop
.text:0808EF49                 nop
.text:0808EF4A                 nop
.text:0808EF4B                 nop
.text:0808EF4C                 nop
.text:0808EF4D                 nop
.text:0808EF4E                 nop
.text:0808EF4F                 mov     al, [ebx+1]
.text:0808EF52                 sub     al, 31h ; '1'
.text:0808EF54                 jnz     short loc_808EFC1
.text:0808EF56                 jmp     near ptr dword_8076E3C+1FDh
.text:0808EF5B ; ---------------------------------------------------------------------------
.text:0808EF5B                 nop
.text:0808EF5C                 nop
.text:0808EF5D                 nop
.text:0808EF5E                 nop
.text:0808EF5F                 nop
.text:0808EF60                 nop
.text:0808EF61                 nop
.text:0808EF62                 nop
.text:0808EF63                 nop
.text:0808EF64                 nop
.text:0808EF65                 nop
.text:0808EF66                 nop
.text:0808EF67                 nop
.text:0808EF68                 nop
.text:0808EF69                 nop
.text:0808EF6A                 nop
.text:0808EF6B                 nop
.text:0808EF6C                 nop
.text:0808EF6D                 mov     al, [ebx+9]
.text:0808EF70                 sub     al, 66h ; 'f'
.text:0808EF72                 jnz     short loc_808EFC1
.text:0808EF74                 jmp     near ptr dword_807563C+5D5h
.text:0808EF79 ; ---------------------------------------------------------------------------
.text:0808EF79                 nop
.text:0808EF7A                 nop
```

Mình thử code lại trong python và thành công:

tmp = [0]*100

tmp[10] = *0x*6c

```
tmp[0x11] = 0x4e

tmp[6] = ord('_')

tmp[0x12] = 0x39

tmp[3] = 0x70

tmp[0xd] = 0x31

tmp[0xf] = ord('n')

tmp[0xb] = ord('a')

tmp[8] = ord('3')

tmp[0xc] = 0x31

tmp[0] = 0x53

tmp[7] = 0x44

tmp[4] = 0x6c

tmp[0x10] = 0x69

tmp[0xe] = 0x65

tmp[2] = 0x6d

tmp[1] = 0x31

tmp[9] = ord('f')

tmp[5] = ord('3')

print(bytes(tmp))
```

```
b'S1mpl3_D3fla11eniN9\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00'
```