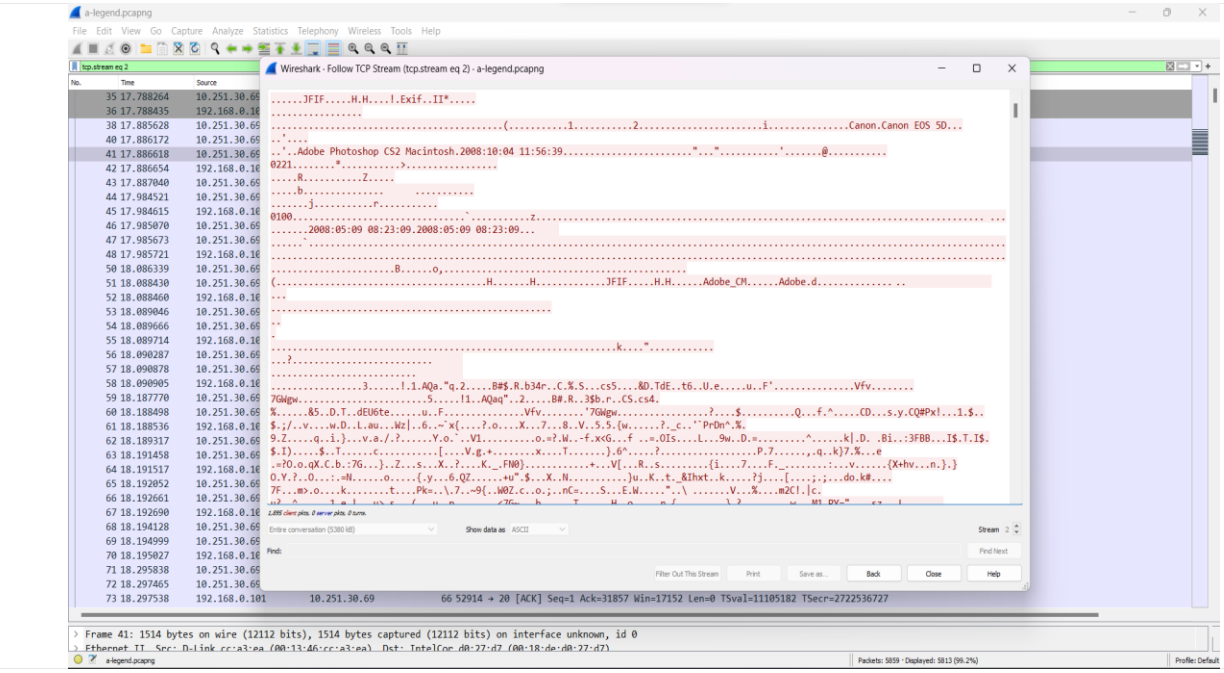
Nguyễn Đình Hải _ B21DCAT004

WRITE UP ĐỘI TUYỂN SVATTT PTIT

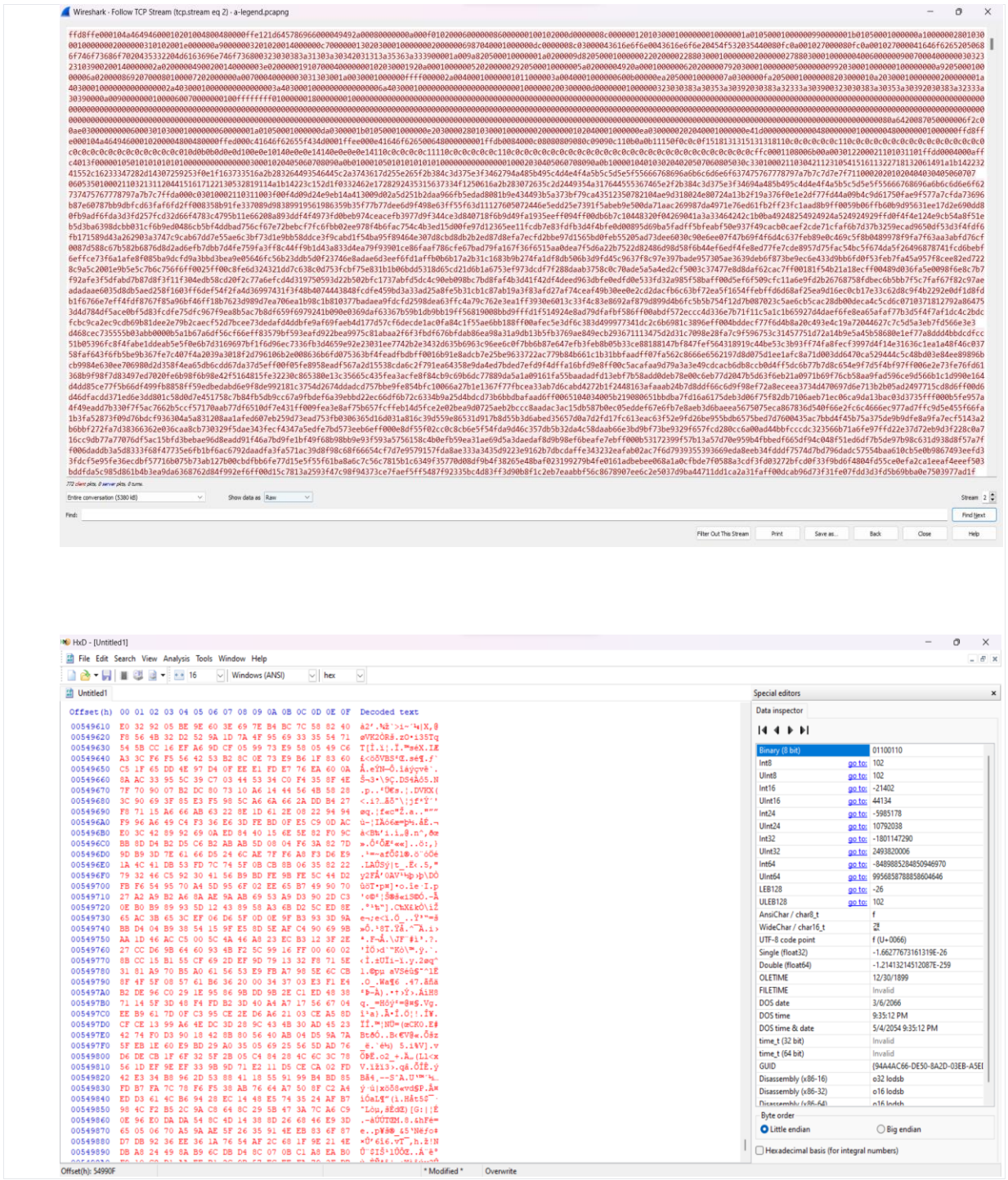
Forensics

A Legend

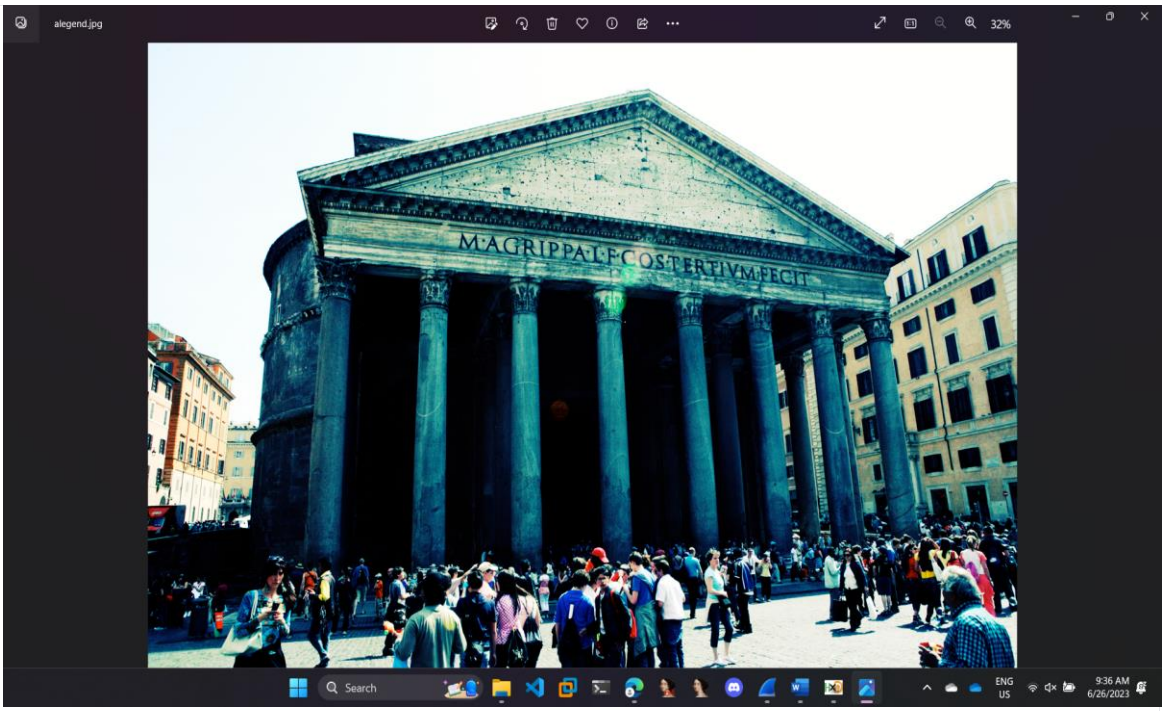
- Sau khi giải nén file ta nhận được 1 file zip và 1 file pcap. Khi check thử zip xong thì thấy đây là 1 file zip có mật khẩu và chúng ta không nhận được mật khẩu này từ để bài. Vậy nhiệm vụ của ta sẽ là tìm kiếm mật khẩu để extract ra được file pass.txt.

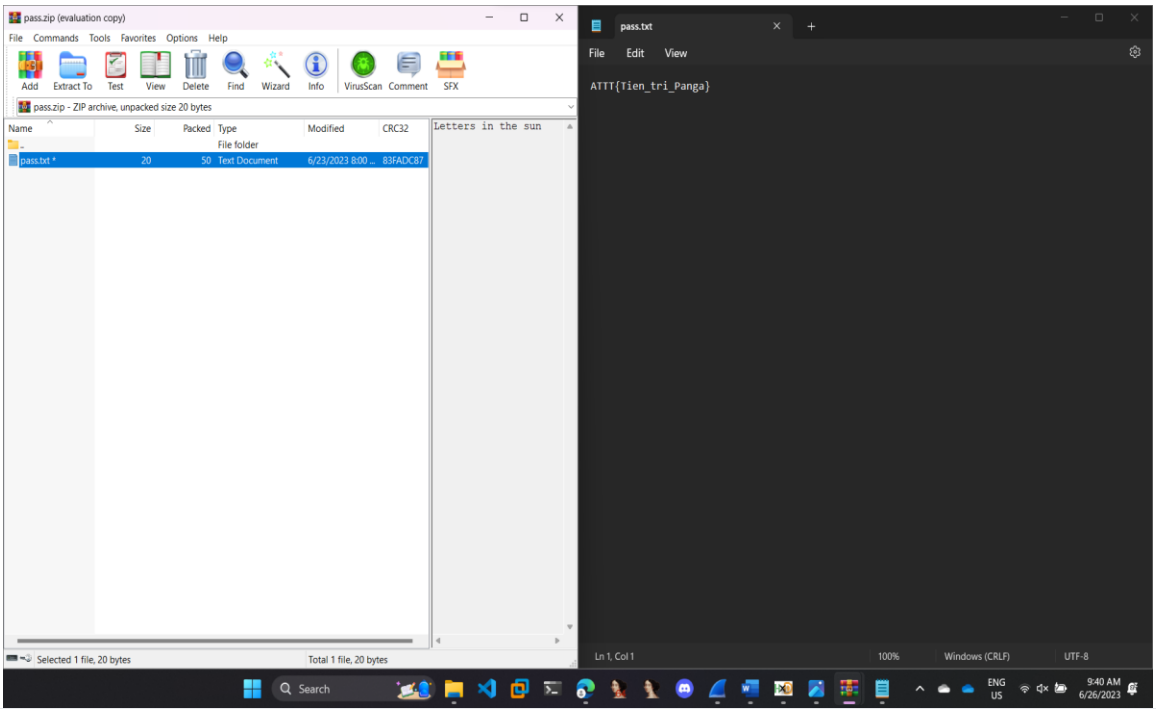


- Sau khi xem ta có thể nhận ra đây chính là file jpg. Vì vậy ta sẽ sử dụng HXD để xem ảnh đề bài cho.
- Chuyển nó sang dạng raw và copy sau đó paste sang HXD và lưu lại vào thành file jpg.



- Đây là ảnh mà ta nhận được. Sau khi xem xét dòng chữ trên tường của bức ảnh và thử với file zip thì nhận được "CO" là mật khẩu của file pass.zip, ngoài ra vệt sáng ở chỗ CO cũng là 1 dấu hiệu cho mật khẩu của file.



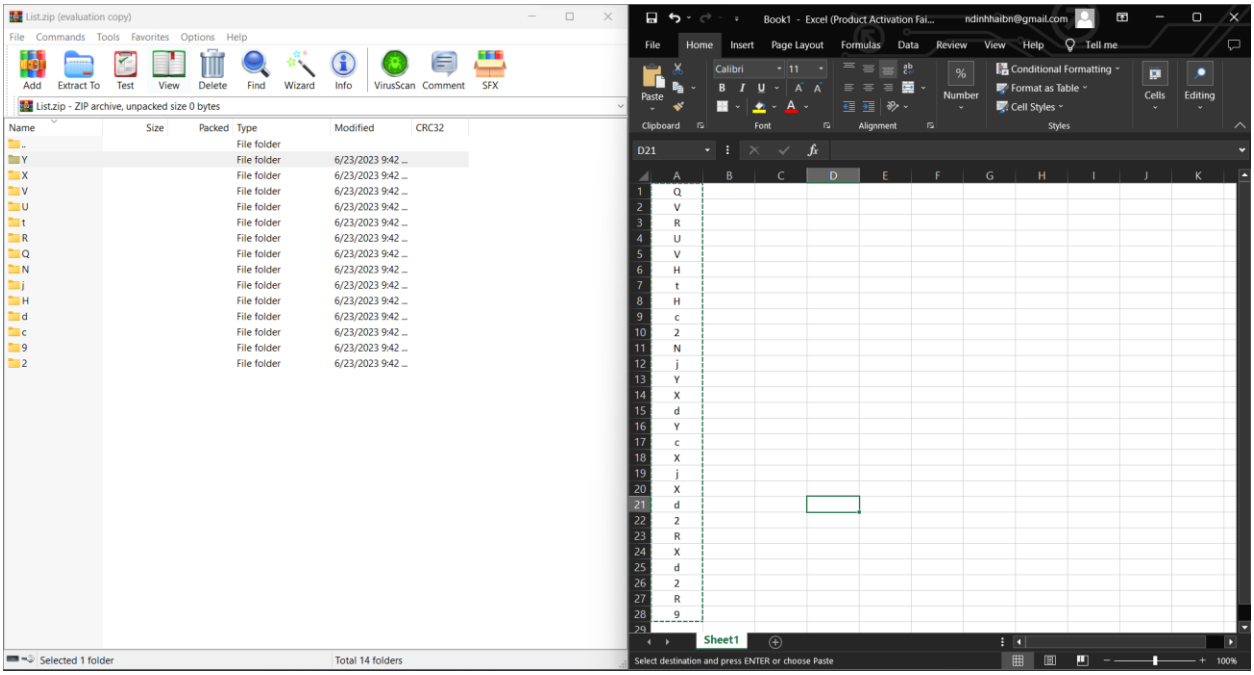


Flag

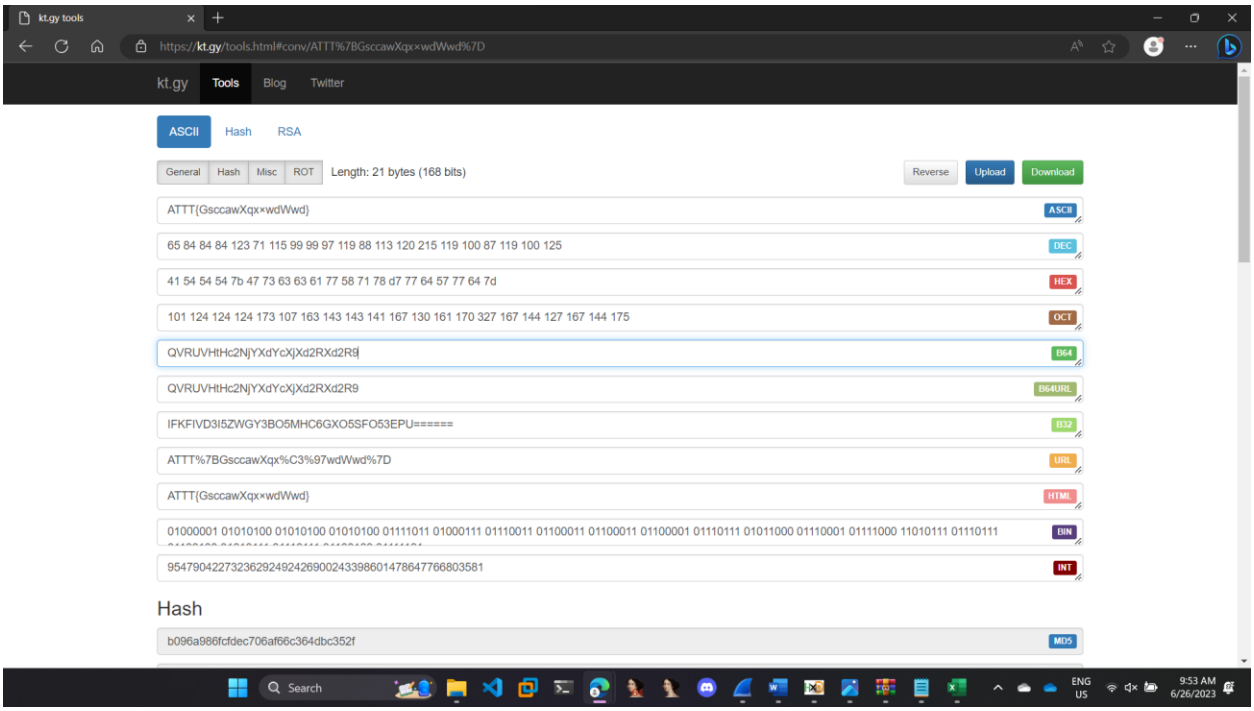
- Chúng ta sẽ có flag: `ATTT{Tien_tri_Panga}`

List

- Ta nhận được từ đề bài 1 file zip và có thể nhận ra các số thứ tự của file trong folder lần lượt là các chữ cái trong chuỗi cần tìm vì sau khi lướt qua ta có thể thấy đây hoàn toàn là các folder và file rỗng.



- Sau khi decode sang từ base64 ta thu được flag cần tìm.



Flag

- Chúng ta sẽ có flag: `ATTT{GsccawXqx×wdWwd}`

Crypto

Cry2

- Đây là đoạn code của đề bài:

```
from Crypto.Util.number import bytes_to_long, getPrime
import random

FLAG = b'FLAG{??????????????????????????????????????????????????}'


def gen_params():
    p = getPrime(1024)
    g = random.randint(2, p - 2)
    x = random.randint(2, p - 2)
    h = pow(g, x, p)
    return (p, g, h), x


def encrypt(pubkey):
    p, g, h = pubkey
    m = bytes_to_long(FLAG)
    y = random.randint(2, p - 2)
    s = pow(h, y, p)
    return (g * y % p, m * s % p)


def main():
    pubkey, _ = gen_params()
    c1, c2 = encrypt(pubkey)

    with open('out.txt', 'w') as f:
        f.write(
            f'p = {pubkey[0]}\ng = {pubkey[1]}\nh = {pubkey[2]}\n(c1, c2) = ({c1},
{c2})\n'
        )


if __name__ == "__main__":
    main()
```

- Nhìn vào các biến ở file output ta sẽ thấy đc các mối liên hệ của các biến

```
p =
140513998383733505878882484463832810400773610216464396082090299095443413923172235117457913756664550823719549662902317573376081431193542463845487697907226369167600481120163289426938922732100465539059713541309125384262046419688008790735321015697953751228026907656116801742757145900027205145531958711127892554959
g =
655569063352527394921387921655423107298472746042154912043776598594879418833318773248286680149452056885395386253477912454340174568672110452746325159033540319877517814437707059453605804644254578285779764575872487038771109795269964514872699920530168549143808455788859903718334974361034866015093296260840839686608
h =
152735287338183640889448255649314165802468048285515114081853860765508449140612473864029766308783590775508127610244142933576472841083221958744316187053184115815018043083472882966852981014282427542705840566868893626883140676007251850450559405035368638949108121914413699713937314560331800826895786912824896309751
(c1, c2) =
(329307631376575912456716826610653789078761574007369632470686092417415141381389954562826773792395418251835558030848258288439508397885562701439901637905429348825340222539095428047144871645250800540911128457674569053203176959512248947733831294361486181395171711027178647852158471339157160205412776372561798181,
767000164159911323186194198034532284709279627624523461702295233536585247715722643201012390732899052818152244387012048014813585252171555002076770594763665053144095878835105615644469858725564738004696588972822443746634263002488094079218868067047363036933846394281034692887669103848918824820201928910120876233220)

Liên hệ:

c1 =
329307631376575912456716826610653789078761574007369632470686092417415141381389954562826773792395418251835558030848258288439508397885562701439901637905429348825340222539095428047144871645250800540911128457674569053203176959512248947733831294361486181395171711027178647852158471339157160205412776372561798181
g =
655569063352527394921387921655423107298472746042154912043776598594879418833318773248286680149452056885395386253477912454340174568672110452746325159033540319877517814437707059453605804644254578285779764575872487038771109795269964514872699920530168549143808455788859903718334974361034866015093296260840839686608
p =
140513998383733505878882484463832810400773610216464396082090299095443413923172235117457913756664550823719549662902317573376081431193542463845487697907226369167600481120163289426938922732100465539059713541309125384262046419688008790735321015697953751228026907656116801742757145900027205145531958711127892554959

c1 = g * y % p

=> tìm được  y

h =
152735287338183640889448255649314165802468048285515114081853860765508449140612473864029766308783590775508127610244142933576472841083221958744316187053184115815018043083472882966852981014282427542705840566868893626883140676007251850450559405035368638949108121914413699713937314560331800826895786912824896309751
s = pow(h, y_value, p)
```

```
c2 =
767000164159911323186194198034532284709279627624523461702295233536585247715722643201012390732899052818152244387012048014813585252171555002076770594763665053144095878835106156444698587255647380046965889728224437466342630024880940792188680670473630369338463942810346928876691038489188248202019289101208762332 2

c2 = m * s % p

=> tìm được m
=> Flag
```

- Để thuận tiện cho việc tìm những số lớn và tính toán trong các biểu thức ta sử dụng z3solver để tìm ra giá trị của y từ đó có thể tìm m. Đây là script để tìm ra flag.

```
from z3 import *
from Crypto.Util.number import *
c1 =
329307631376575912456716826610653789078761574700736963247068609241741514138138995456282677379239541825183555803084825828843950839788556270143990163790542934882534022253909542804714487164525080054091112845767456905320317695951224894773383129436148618139517171102717864785215847133915716020541277637256179818 11
g =
655569063352527394921387921655423107298472746042154912043776598594879418833318773248286680149452056885395386253477912454340174568672110452746325159033540319877517814437707059453605804644254578285779764575872487038771109795269964514872699920530168549143808455788859903718334974361034866015093296260840839686 60
p =
140513998383733505878882484463832810400773610216464396082090299095443413923172235117457913756664550823719549662902317573376081431193542463845487697907226369167600481120163289426938922732100465539059713541309125384262046419688008790735321015697953751228026907656116801742757145900027205145531958711127892554959

solver = Solver()

y = Int('y')

solver.add(c1 == g * y % p)
solver.add(y > 0)

if solver.check() == sat:
    model = solver.model()
    y_value = model[y].as_long()
    print("y = ", y_value)
else:
    print("ko co")

solver = Solver()
```

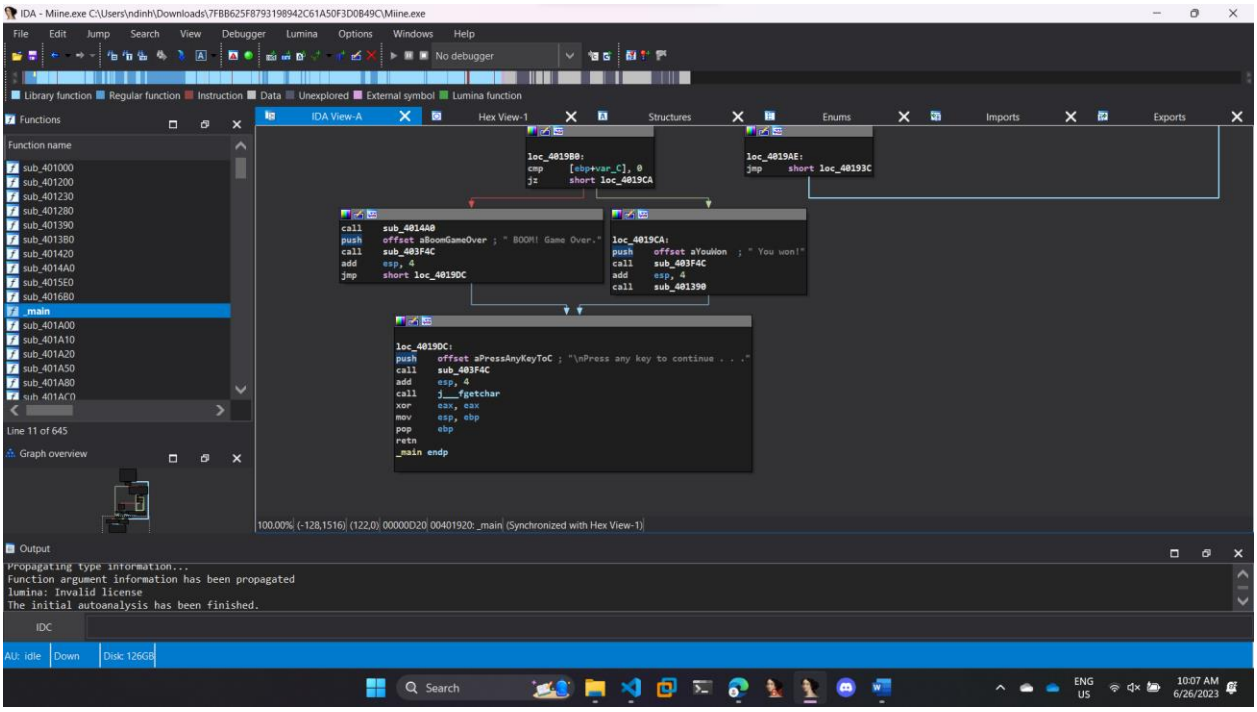
```
h =
152735287338183640889448255649314165802468048285515114081853860765508449140612473864029766308783590775508127610244142933576472841083221958744316187053184115815018043083472882966852981014282427542705840566868893626883140676007251850450559405035368638949108121914413699713937314560331800826895786912824896309750
s = pow(h, y_value, p)
c2 =
767000164159911323186194198034532284709279627624523461702295233536585247715722643201012390732899052818152244387012048014813585252171555002076770594763665053144095878835106156444698587255647380046965889728224437466342630024880940792188680670473630369338463942810346928876691038489188248202019289101208762332 2
m = Int('m')

solver.add(c2 == m * s % p)
solver.add(m > 0)


if solver.check() == sat:
    model = solver.model()
    m_value = model[m].as_long()
    print(long_to_bytes(m_value))
else:
    print("ko co")
```

Flag

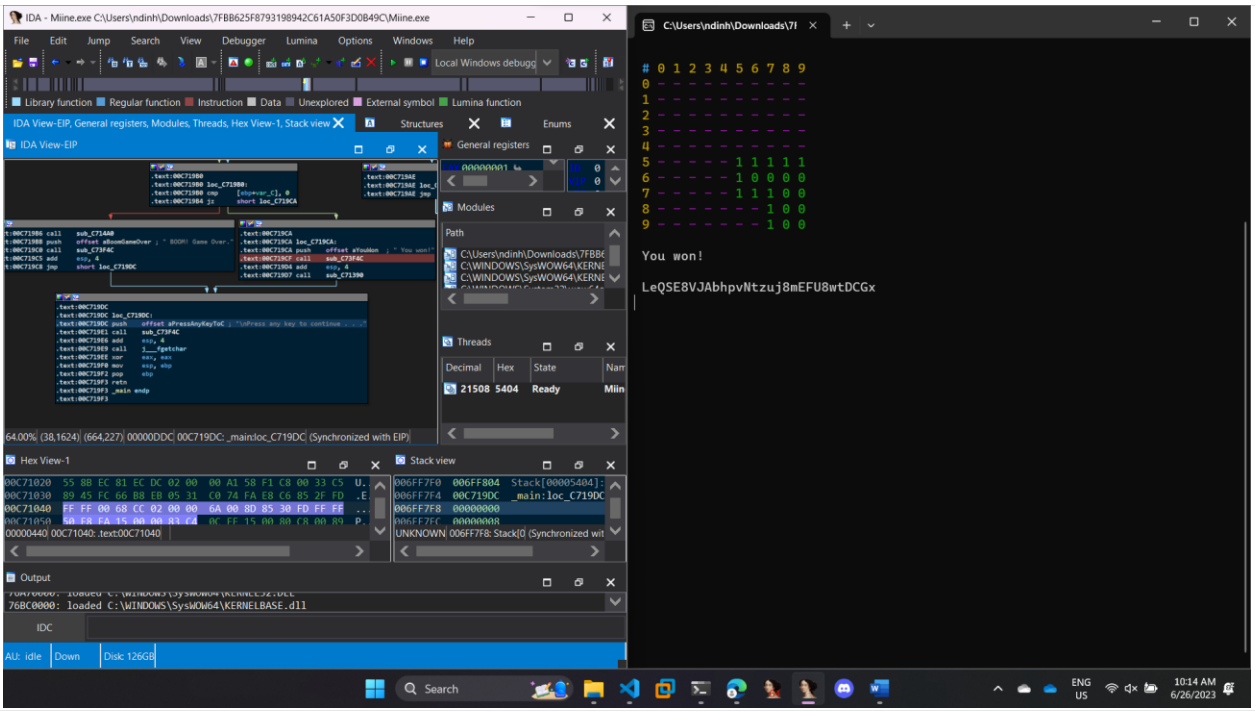
```
 Halston  {  \Downloads\7FBB625F8793198942C61A50F3D0B49C  55ms  10:03 AM 
 ndinh ❯❯ python -u
"c:\Users\ndinh\Downloads\7FBB625F8793198942C61A50F3D0B49C\s.py"
y =
825593139401551717241004249831746011611220933629723971167457805557091408044274609872135566522528463194640897645334285292361566952799893572771956984923449992035038645143710674968921194934630524580648819081510823036480372084403926146118277106673447483658551140654066075345925633209371725712680285995931995990121
b'FLAG{s0me_m4th_1s_3asy_1f_y0u_kn0w_4b0ut_m0dular_4r1thm3t1c}'
```

Reverse

Re1



- Sau khi check qua luồng thực thi ta sẽ nhìn thấy được hàm win và hiểu được rằng ta sẽ nhận được 1 chuỗi Str sau đó sẽ được build và decode qua 1 đoạn loằng ngoằng sẽ cho ra 1 chuỗi win. Trước tiên ta sẽ thử patch đến nó xem ta nhận được những gì. Các bước patch em sẽ skip qua vì chỉ cần call đến các hàm cần gọi điều hướng đến hàm win. Sau khi nhảy đến đó ta nhận được chuỗi như sau.



- Sau khi decode qua base58 dưới dạng Ripple ta thu được chuỗi `Happy_Minesweeper_Game` và thử sub với flag `ATTT{Happy_Minesweeper_Game}` nhưng không đúng sau khi đọc kĩ lại code và thử vận may của mình em đã sub thử với flag khác là `ATTT{H4ppy_M1n35w33p3r_G4m3}` nhưng vẫn không đúng.

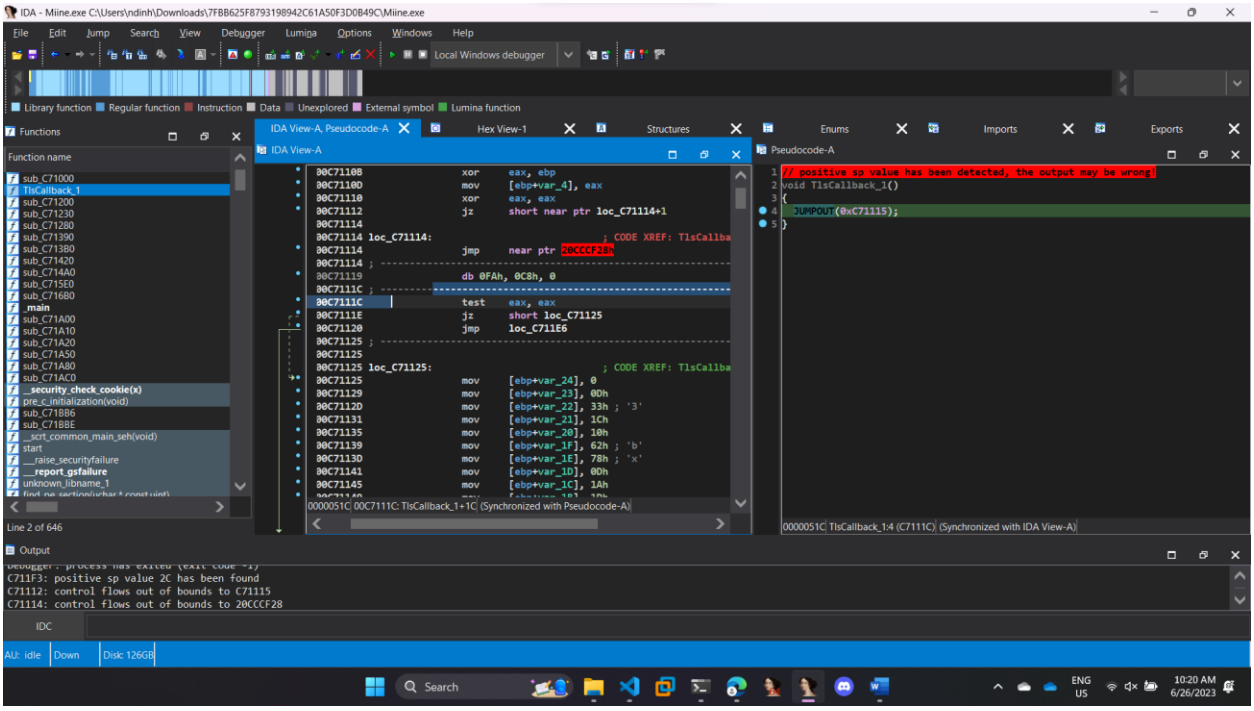
- Sau khi xem qua thì có thể thấy hàm TlsCallback_1 này sẽ xor các giá trị của chuỗi Str với 1 chuỗi nào đó. Vì vậy ta sẽ tạo function để check.

```
 public TlsCallback_1
.text:00C71100 TlsCallback_1   proc near               ; DATA XREF: .rdata:00C88150↓o
.text:00C71100
.text:00C71100 var_2C          = dword ptr -2Ch
.text:00C71100 var_28          = dword ptr -28h
.text:00C71100 var_24          = byte ptr -24h
.text:00C71100 var_23          = byte ptr -23h
.text:00C71100 var_22          = byte ptr -22h
.text:00C71100 var_21          = byte ptr -21h
.text:00C71100 var_20          = byte ptr -20h
.text:00C71100 var_1F          = byte ptr -1Fh
.text:00C71100 var_1E          = byte ptr -1Eh
.text:00C71100 var_1D          = byte ptr -1Dh
.text:00C71100 var_1C          = byte ptr -1Ch
.text:00C71100 var_1B          = byte ptr -1Bh
```

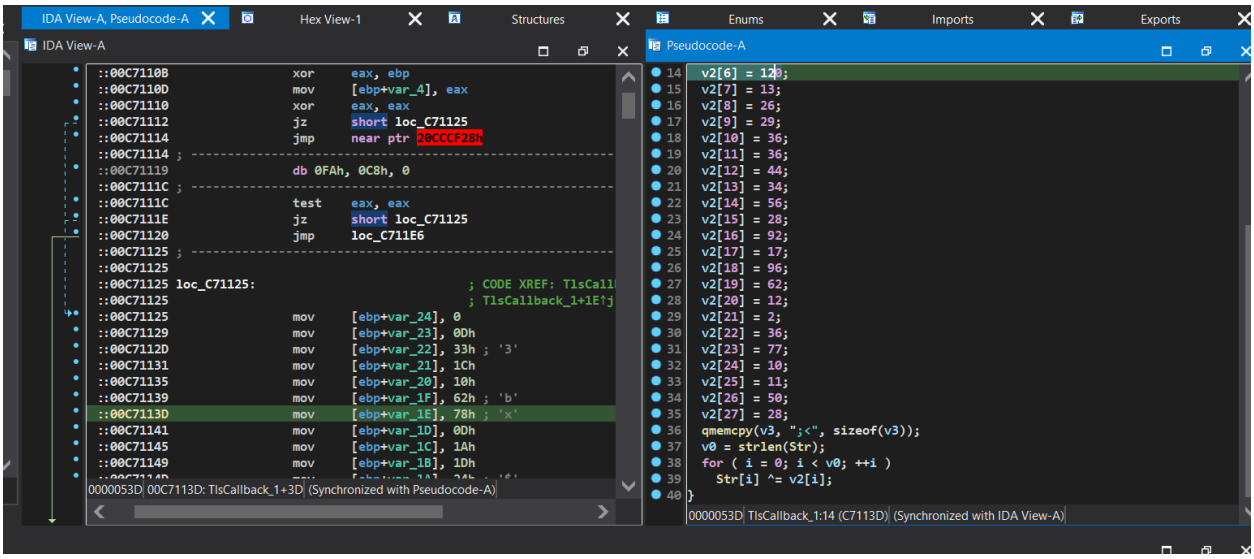
```
.text:00C71100 var_1A          = byte ptr -1Ah
.text:00C71100 var_19          = byte ptr -19h
.text:00C71100 var_18          = byte ptr -18h
.text:00C71100 var_17          = byte ptr -17h
.text:00C71100 var_16          = byte ptr -16h
.text:00C71100 var_15          = byte ptr -15h
.text:00C71100 var_14          = byte ptr -14h
.text:00C71100 var_13          = byte ptr -13h
.text:00C71100 var_12          = byte ptr -12h
.text:00C71100 var_11          = byte ptr -11h
.text:00C71100 var_10          = byte ptr -10h
.text:00C71100 var_F           = byte ptr -0Fh
.text:00C71100 var_E           = byte ptr -0Eh
.text:00C71100 var_D           = byte ptr -0Dh
.text:00C71100 var_C           = byte ptr -0Ch
.text:00C71100 var_B           = byte ptr -0Bh
.text:00C71100 var_A           = byte ptr -0Ah
.text:00C71100 var_9           = byte ptr -9
.text:00C71100 var_8           = byte ptr -8
.text:00C71100 var_7           = byte ptr -7
.text:00C71100 var_4           = dword ptr -4
.text:00C71100
.text:00C71100                 push    ebp
.text:00C71101                 mov     ebp, esp
.text:00C71103                 sub     esp, 2Ch
.text:00C71106                 mov     eax, ___security_cookie
.text:00C7110B                 xor     eax, ebp
.text:00C7110D                 mov     [ebp+var_4], eax
.text:00C71110                 xor     eax, eax
.text:00C71112                 jz      short near ptr loc_C71114+1
.text:00C71114
.text:00C71114 loc_C71114:                             ; CODE XREF:
TlsCallback_1+12↑j
.text:00C71114                 jmp     near ptr 20CCCF28h
.text:00C71114 ; ---------------------------------------------------------------------
-------
.text:00C71119                 db 0FAh, 0C8h, 0
.text:00C7111C ; ---------------------------------------------------------------------
-------
.text:00C7111C                 test    eax, eax
.text:00C7111E                 jz      short loc_C71125
.text:00C71120                 jmp     loc_C711E6
.text:00C71125 ; ---------------------------------------------------------------------
-------
.text:00C71125
.text:00C71125 loc_C71125:                             ; CODE XREF:
TlsCallback_1+1E↑j
.text:00C71125                 mov     [ebp+var_24], 0
.text:00C71129                 mov     [ebp+var_23], 0Dh
.text:00C7112D                 mov     [ebp+var_22], 33h ; '3'
.text:00C71131                 mov     [ebp+var_21], 1Ch
.text:00C71135                 mov     [ebp+var_20], 10h
.text:00C71139                 mov     [ebp+var_1F], 62h ; 'b'
.text:00C7113D                 mov     [ebp+var_1E], 78h ; 'x'
```

```
.text:00C71141                 mov     [ebp+var_1D], 0Dh
.text:00C71145                 mov     [ebp+var_1C], 1Ah
.text:00C71149                 mov     [ebp+var_1B], 1Dh
.text:00C7114D                 mov     [ebp+var_1A], 24h ; '$'
.text:00C71151                 mov     [ebp+var_19], 24h ; '$'
.text:00C71155                 mov     [ebp+var_18], 2Ch ; ','
.text:00C71159                 mov     [ebp+var_17], 22h ; '"'
.text:00C7115D                 mov     [ebp+var_16], 38h ; '8'
.text:00C71161                 mov     [ebp+var_15], 1Ch
.text:00C71165                 mov     [ebp+var_14], 5Ch ; '\'
.text:00C71169                 mov     [ebp+var_13], 11h
.text:00C7116D                 mov     [ebp+var_12], 60h ; '`'
.text:00C71171                 mov     [ebp+var_11], 3Eh ; '>'
.text:00C71175                 mov     [ebp+var_10], 0Ch
.text:00C71179                 mov     [ebp+var_F], 2
.text:00C7117D                 mov     [ebp+var_E], 24h ; '$'
.text:00C71181                 mov     [ebp+var_D], 4Dh ; 'M'
.text:00C71185                 mov     [ebp+var_C], 0Ah
.text:00C71189                 mov     [ebp+var_B], 0Bh
.text:00C7118D                 mov     [ebp+var_A], 32h ; '2'
.text:00C71191                 mov     [ebp+var_9], 1Ch
.text:00C71195                 mov     [ebp+var_8], 3Bh ; ';'
.text:00C71199                 mov     [ebp+var_7], 3Ch ; '<'
.text:00C7119D                 push    offset Str      ;
"LcKUC8JBInrfjVzdeb8qCHE8ozXSMt"
.text:00C711A2                 call    _strlen
.text:00C711A7                 add     esp, 4
.text:00C711AA                 mov     [ebp+var_2C], eax
.text:00C711AD                 mov     [ebp+var_28], 0
.text:00C711B4                 jmp     short loc_C711BF
.text:00C711B6 ; ---------------------------------------------------------------------
-------
.text:00C711B6
.text:00C711B6 loc_C711B6:                             ; CODE XREF:
TlsCallback_1+E4↓j
.text:00C711B6                 mov     ecx, [ebp+var_28]
.text:00C711B9                 add     ecx, 1
.text:00C711BC                 mov     [ebp+var_28], ecx
.text:00C711BF
.text:00C711BF loc_C711BF:                             ; CODE XREF:
TlsCallback_1+B4↑j
.text:00C711BF                 mov     edx, [ebp+var_28]
.text:00C711C2                 cmp     edx, [ebp+var_2C]
.text:00C711C5                 jge     short loc_C711E6
.text:00C711C7                 mov     eax, [ebp+var_28]
.text:00C711CA                 movsx   ecx, [ebp+eax+var_24]
.text:00C711CF                 mov     edx, [ebp+var_28]
.text:00C711D2                 movsx   eax, byte ptr Str[edx] ;
"LcKUC8JBInrfjVzdeb8qCHE8ozXSMt"
.text:00C711D9                 xor     eax, ecx
.text:00C711DB                 mov     ecx, [ebp+var_28]
.text:00C711DE                 mov     byte ptr Str[ecx], al ;
"LcKUC8JBInrfjVzdeb8qCHE8ozXSMt"
.text:00C711E4                 jmp     short loc_C711B6
```

```
.text:00C711E6 ; ---------------------------------------------------------------------------
-------
.text:00C711E6
.text:00C711E6 loc_C711E6:                             ; CODE XREF:
TlsCallback_1+20↑j
.text:00C711E6                                         ; TlsCallback_1+C5↑j
.text:00C711E6                 mov     ecx, [ebp+var_4]
.text:00C711E9                 xor     ecx, ebp        ; StackCookie
.text:00C711EB                 call    @__security_check_cookie@4 ;
__security_check_cookie(x)
.text:00C711F0                 mov     esp, ebp
.text:00C711F2                 pop     ebp
.text:00C711F3                 retn    0Ch
.text:00C711F3 TlsCallback_1   endp
```



- Tại đây có thể thấy nó bị jumpout ngay khi bắt đầu khởi tạo biến và stack. Ta sẽ patch qua để cho nó tiếp tục thực hiện luồng code tiếp theo của mình.



- Sau khi patch xong ta đọc source c thì có thể thấy biến Str sẽ được thay đổi khi xor với v2.

```
#include <bits/stdc++.h>

using namespace std;

int main(){

  signed int v0; // [esp+0h] [ebp-2Ch]
  signed int i; // [esp+4h] [ebp-28h]
  char v2[28]; // [esp+8h] [ebp-24h]
  char Str[] = "LcKUC8JBInrfjVzdeb8qCHE8ozXSMt";
  v2[0] = 0;
  v2[1] = 13;
  v2[2] = 51;
  v2[3] = 28;
  v2[4] = 16;
  v2[5] = 98;
  v2[6] = 120;
  v2[7] = 13;
  v2[8] = 26;
  v2[9] = 29;
  v2[10] = 36;
  v2[11] = 36;
  v2[12] = 44;
  v2[13] = 34;
  v2[14] = 56;
  v2[15] = 28;
  v2[16] = 92;
  v2[17] = 17;
  v2[18] = 96;
```

```
  v2[19] = 62;
  v2[20] = 12;
  v2[21] = 2;
  v2[22] = 36;
  v2[23] = 77;
  v2[24] = 10;
  v2[25] = 11;
  v2[26] = 50;
  v2[27] = 28;
  v0 = strlen(Str);
  for ( i = 0; i < v0; ++i )
    Str[i] ^= v2[i];

  cout << Str;

    return 0;
}
 Halston  {  \Downloads\7FBB625F8793198942C61A50F3D0B49C  53ms  10:25 AM 
 ndinh ❯❯ cd "c:\Users\ndinh\Downloads\7FBB625F8793198942C61A50F3D0B49C\" ; if
($?) { g++ t.cpp -o t } ; if ($?) { .\t }
LnxISZ2OSsVBFtBx9sXOOJaueqjOMt
```

- Cầm chuỗi trên để decrypt với hàm win với đoạn code sau.

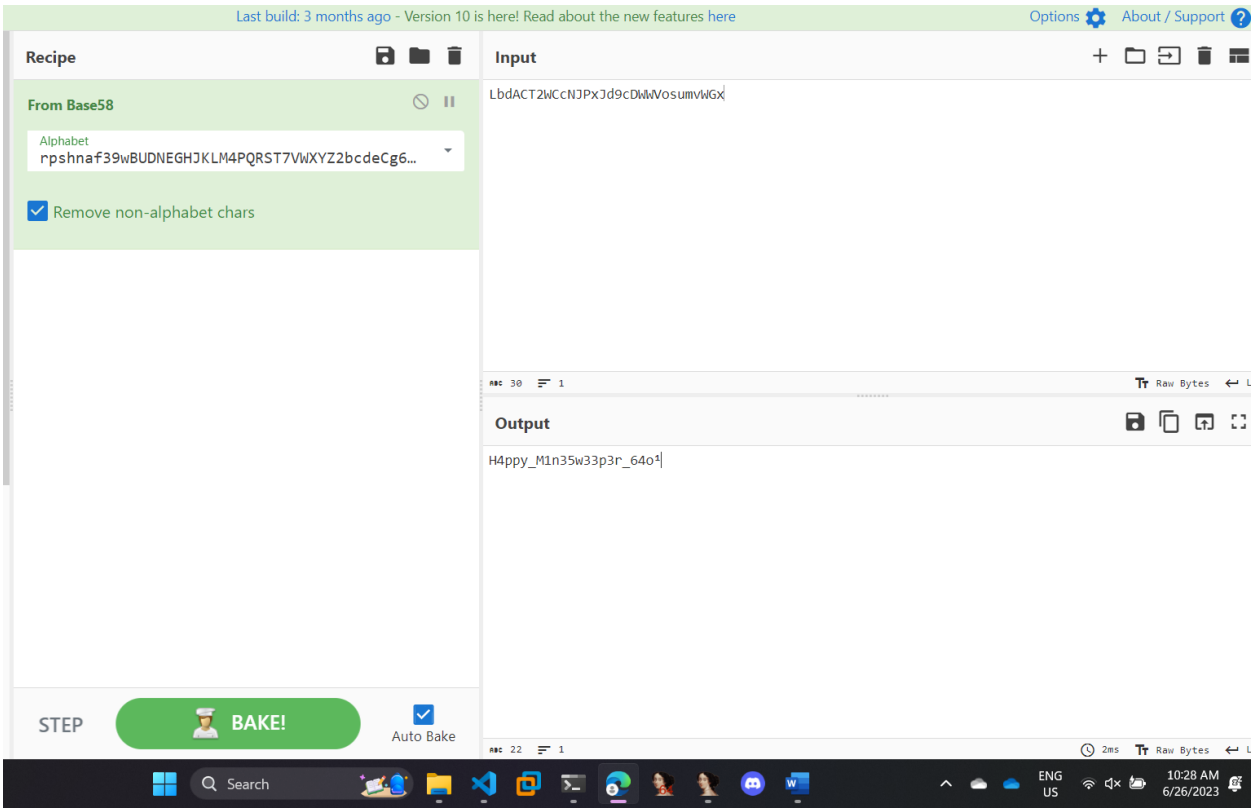
```
#include <bits/stdc++.h>

using namespace std;

int check(int a1, int a2){
    int v4 = a1 % a2;
    for (int i = 1; i < a2; ++i )
    {
        if ( i * v4 % a2 == 1 )
        return i;
    }

    return -1;
}

int main(){
    char s[] = "LnxISZ2OSsVBFtBx9sXOOJaueqjOMt";
    int a2 = 5;
    char flag[100];
    int a3 = 8;
    int cnt = strlen(s) + 1;
    int x = 26;
    int v3 = check(a2, x);
    for( int i = 0; i < cnt; i++){
        int v7 = s[i];
```

```
        if ( v7 < 65 || v7 > 90 )
        {
        if ( v7 >= 97 && v7 <= 122 )
            v7 = v3 * ((v7 - 97 - a3 + 26) % 26) % 26 + 97;
        }
        else
        {
        v7 = v3 * ((v7 - 65 - a3 + 26) % 26) % 26 + 65;
        }
        flag[i] = v7;
    }
    cout << flag;
    return 0;
}
```

- Ta thu được chuỗi base58 mới.

```
 Halston  {  \Downloads\7FBB625F8793198942C61A50F3D0B49C  2.129s  10:25 AM
 
 ndinh ❯❯ cd "c:\Users\ndinh\Downloads\7FBB625F8793198942C61A50F3D0B49C\" ; if
($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
.\tempCodeRunnerFile }
LbdACT2WCcNJPxJd9cDWWVosumvWGx
```

• Decode nó với base58 ripple thì có thể thấy 2 byte cuối bị lem và sau khi nhìn qua các chữ khác cũng có thể thấy byte cuối sẽ là `3` còn byte kề nó sẽ là `m`



Flag

- Sau khi thử sub ta thu được flag `ATTT{H4ppy_M1n35w33p3r_64m3}` thì đã correct. Khá là cay khi lần thử đầu tiên chỉ sai 1 kí tự của FLAG 😣