Crypto: [hgame2024-week3] exRSA(三元维纳扩展攻击,LLL)

Code：

```
from Crypto.Util.number import *
from secret import flag
m=bytes_to_long(flag)
p=getStrongPrime(1024)
q=getStrongPrime(1024)
phi=(p-1)*(q-1)
e1=inverse(getPrime(768),phi)
e2=inverse(getPrime(768),phi)
e3=inverse(getPrime(768),phi)
n=p*q
c=pow(m,0x10001,n)
print(f'e1={e1}')
print(f'e2={e2}')
print(f'e3={e3}')
print(f'c={c}')
print(f'n={n}')

"""
e1=507704823781196942747311122537087612252896744705655189912361346179268800289678839430419291761056414976625223228157699029348523968414531087693099791896007081696882915037687595340542080958626715317171749619833686108952370183209832228450193114288981757581676170504495170553084932792884984815864303069336314375706322058471492589396558796704213755780726115411791635851947796464529347197506336205069030635362749298086100843976536583762265797795806985328805630725316750988325812294988227702166531780725330890635567047217234617117726768806495939718692610398725955158662796540697911819348552752097674849072846016794905528953940697911819348552752097674849072846016794905528953e2=125268482983490053905202769239291324634591525749986257572082592978911151336541176482157829453325290813652738603162011307933065707777350765347721689997058956412075353038394550740030576878103811109783209889760113261069199407991609742283118247600463702735055110656192685576971825862592343792394104827844498157323352943956763022264168637093400329876127151519160842918210954626258210231335604153258248853472213914969372132463617363612708467411285575956030527136125284537099484031007112776796412185204298788975656554820864105763799714047892122976975537482924381830655009933750400317338254966927976993624210102715995102694014e3=129859407575785308105193703320636583440466888566059674749410144368727203604440404646447909809769913939709470233983574222038732842948434011440650139114636705015598886011451086519610983482508241666976655284176683
```

74408814572959722789020110396245076275553505878565603509466220710219260
03778384927647539728342106871608863818699477815354281768196305958165110
35635788041451561575843367126788829956856326156868539801760476833269742
83896343322981521150211317597571554542488921290158122634140571148036732
89380806411904832885513405470912087789594167016642166480618671034682449
40547830257334758980812478248879675504185090382762 79
c=14141760601523018421104970980245971892462591720193354149001274520982 3
94304182592602851743707531629494335532394745892801055691290913973928 29
24255506647305696872907898950473108556417350199783145349691087255926287
36328692201184114333953086330019823923149070739338307617479181899415881
58573919308029362804475888084406074153773913366045334400997938492378572
47557582307391329320515996021820000355560514217505643587026994918588311
12714356685803665331598517755196383642972851574564680712363719325985985
66304521551389866102720674802573305921461351081900835788730941331144400
50860844192259441093236787002715737932342847147399
n=17853303733838066173110417890593704464146824886316456780873352559969
74261575529446666443952935271843439955281863535276803353194800973717069
75662868487108328004263113285609241336984816535940077278770315062657063
41560810588064209681809146597572126173303463125668183837840427667101827
23475282374748379294453689307018801035764447851214333201478653969853522
01397844403144813714640539547698227384078081619469432167147296858208969
72467020893493349051243983390018762076812868678098172416465691550285372
84640299199579434901583886822168621639659732727311016592278981431585846
20497062552540667240129258151004349538218568545297 53
"""

典型的三元维纳扩展攻击的题

参考 ctf-wiki

https://ctf-wiki.org/crypto/asymmetric/rsa/d_attacks/rsa_extending_wiener/

三个小解密指数的情况

- 对于三个指数的情况我们额外选取$G_{1,3}, W_1G_{2,3}, W_2G_{1,3}$

  这样我们的向量 b 为

$$B = (\ k_1k_2k_3 \quad d_1gk_2k_3 \quad k_1d_2gk_3 \quad d_1d_2g^2k_3 \quad k_1k_2d_3g \quad k_1d_3g \quad k_2d_3g \quad d_1d_2d_3$$

  然后我们便可以构造格

$$L_3 = \begin{pmatrix} 1 & -N & 0 & N^2 & 0 & 0 & 0 & -N^3 \\ 0 & e_1 & -e_1 & -Ne_1 & -e_1 & 0 & Ne_1 & N^2e_1 \\ 0 & 0 & e_2 & -Ne_2 & 0 & Ne_2 & 0 & N^2e_2 \\ 0 & 0 & 0 & e_1e_2 & 0 & -e_1e_2 & -e_1e_2 & -Ne_1e_2 \\ 0 & 0 & 0 & 0 & e_3 & -Ne_3 & -Ne_3 & N^2e_3 \\ 0 & 0 & 0 & 0 & 0 & e_1e_3 & 0 & -Ne_1e_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_2e_3 & -Ne_2e_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_1e_2e_3 \end{pmatrix}$$

  其中

$$D = diag(\ N^{\frac{3}{2}} \quad N \quad N^{a+\frac{3}{2}} \quad \sqrt{N} \quad N^{a+\frac{3}{2}} \quad N^{a+1} \quad N^{a+1} \quad 1\ )$$

  同样我们可以得到

$$\|bL_2\| < \sqrt{8}N^{3/2+2\alpha_3}$$

  则当

$$\alpha_3 < 2/5 - \epsilon^{'}$$

构造如上的矩阵和对角阵，求 B 的格，exp 编写如下

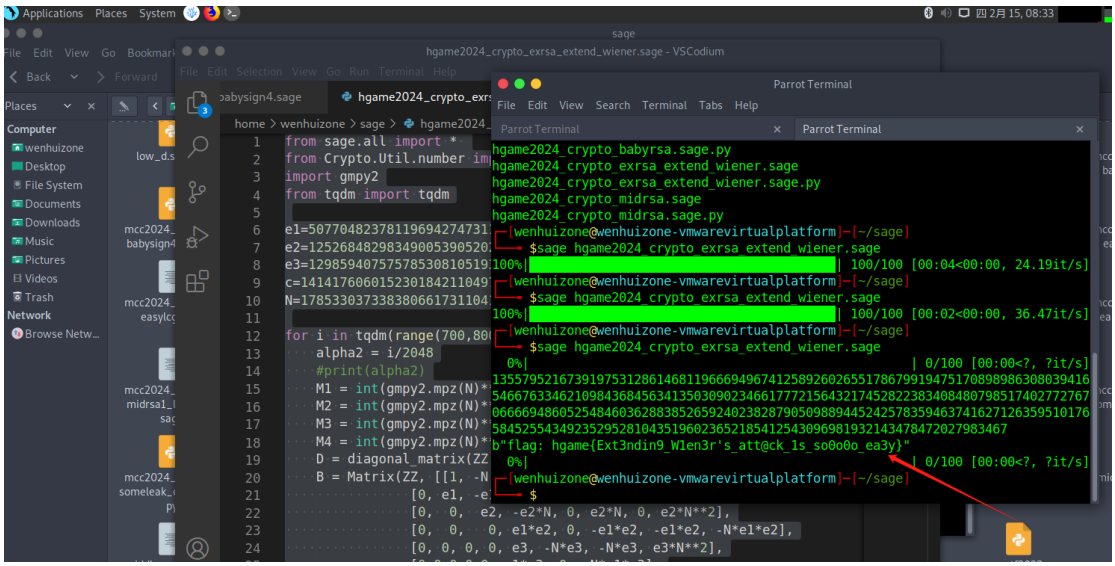
Exp：

```
from sage.all import *
from Crypto.Util.number import *
import gmpy2
from tqdm import tqdm

e1=507704823781196942747311122537087612252896744705655189912361346179268800289678839430419291761056414976625223228157699029348523968414531087693099791896007081696882915037687595340542080958626715317171749619833686108952370183209832228450193114288981757581676170504495170553084932792884984815864303069336314375706322058471492589396558796704213755780726115411791635851947796464529347197506336205069030635362749298086100843976536583762265797795806985328805630725316750988325812294988227702166531780725330890635567047217234617117726768806495939718692610398725955158662796540697911819348552752097674849072846016794905528953 9
e2=125268482983490053905202769239291324634591525749986257572082592978911151336541176482157829453325290813652738603162011307933065707777350765347721689997058956412075353038394550740030576878103811109783209889760113261069199407991609742283118247600463702735055110656192685576971825862592343792394104827844498157323352943956763022264168637093400329876127151519160842918210954626258210231335604153258248853472213914969372132463617363612708467411285575956030527136125284537099484031007112776796412185204298788975656554820864105763799714047892122976975537482924381830655009 9
```

337504003173382549669279769936242101027159951026940 1
e3=129859407575785308105193703320636583440466888566059674749410144368727203604440404646447909809769913939709470233983574222038732842948434011440650139114636705015598886011451086519610983482508241666976655284176683744088145729597227890201103962450762755535058785656035094662207102192600377838492764753972834210687160886381869947781535428176819630595816511035635788041451561575843367126788829956856326156868539801760476833269742838963433229815211502113175975715545424889212901581226341405711480367328938080641190483288551340547091208778959416701664216648061867103468244940547830257334758980812478248879675504185090382762791c=1414176060152301842110497098024597189246259172019335414900127452098233943041825926028517437075316294943355323947458928010556912909139739282924255506647305696872907898950473108556417350199783145349691087255926287363286922011841143339530863300198239231490707393383076174791818994158815857391930802936280447588808440607415377391336604533440099793849237857247557582307391329320515996021820000355560514217505643587026994918588311127143566858036653315985177551963836429728515745646807123637193259859856630452155138986610272067480257330592146135108190083578873094133114440050860844192259441093236787002715737932342847147399N=178533037338380661731104178905937044641468248863164567808733525599697426157552944666644395293527184343995528186353527680335319480097371706975662868487108328004263113285609241336984816535940077278770315062657063415608105880642096818091465975721261733034631256681838378404276671018272347528237474837929445368930701880103576444785121433320147865396985352201397844403144813714640539547698227384078081619469432167147296858208969724670208934933490512439833900187620768128686780981724164656915502853728464029919957943490158388682216862163965973272731101659227898143158584620497062552540667240129258151004349538218568545297531 1

for i in tqdm(range(700,800)):
    alpha2 = i/2048
    #print(alpha2)
    M1 = int(gmpy2.mpz(N)**1.5)
    M2 = int(gmpy2.mpz(N)**(alpha2+1.5))
    M3 = int(gmpy2.mpz(N)**0.5)
    M4 = int(gmpy2.mpz(N)**(alpha2+1))
    D = diagonal_matrix(ZZ, [M1, gmpy2.mpz(N), M2, M3, M2, M4, M4, 1])
    B = Matrix(ZZ, [[1, -N,   0,   N**2, 0, 0, 0, -N**3],
                    [0, e1, -e1, -e1*N, -e1, 0, N*e1, e1*N**2],
                    [0,   0,   e2, -e2*N, 0, e2*N, 0, e2*N**2],
                    [0,   0,   0, e1*e2, 0, -e1*e2, -e1*e2, -N*e1*e2],
                    [0, 0, 0, 0, e3, -N*e3, -N*e3, e3*N**2],
                    [0,0,0,0,0, e1*e3, 0, -N*e1*e3],
                    [0,0,0,0,0,0,e2*e3,-N*e2*e3],

```
                [0,0,0,0,0,0,0,e1*e2*e3]]) * D
    L = B.LLL()
    v = Matrix(ZZ, L[0])
    x = v * B**(-1)
    phi = (x[0,1]/x[0,0]*e1).floor()
    try:
        PR = PolynomialRing(ZZ, 'x')
        x = PR.gen()
        f = x**2 - (N-phi+1)*x + N
        p = f.roots()[0][0]
        print(p)
        q = int(N) // int(p)
    except:
        pass

    #print(phi)
    d = inverse_mod(65537, phi)
    m = power_mod(c, d, N)
    #print(m)
    #print(long_to_bytes(int(m)))
    if b'hgame{' in long_to_bytes(int(m)):
        print(b'flag: ' + long_to_bytes(m))
        break
```



Flag：hgame{Ext3ndin9_W1en3r's_att@ck_1s_so0o0o_ea3y}

Crypto: [hgame2024-week3] matrix_equation（LLL,格基规约）

Code：

```
from Crypto.Util.number import *
import hashlib
from secret import p,q,r
k1=getPrime(256)
k2=getPrime(256)
temp=p*2**256+q*k1+r*k2
hint=len(bin(temp)[2:])
flag='hgame{'+hashlib.sha256(str(p+q+r).encode()).hexdigest()+'}'
print(f'hint={hint}')
print(f'k1={k1}')
print(f'k2={k2}')
"""
83
k1=737153298772153401459512383432471562821657053960747864832566998176512
55709671
k2=613619706622698697382703285238977654084439071983136324100684542237178
24276837
"""
```

已知

$$temp = 2^{256} * p + k_1 * q + k_2 * r$$

构造格：

$$M = \begin{pmatrix} 1 & 0 & 2^{256} \\ 0 & 1 & k_1 \\ 0 & 0 & k_2 \end{pmatrix}$$

使得：

$$\begin{pmatrix} p & q & r \end{pmatrix} M = \begin{pmatrix} p & q & temp \end{pmatrix}$$

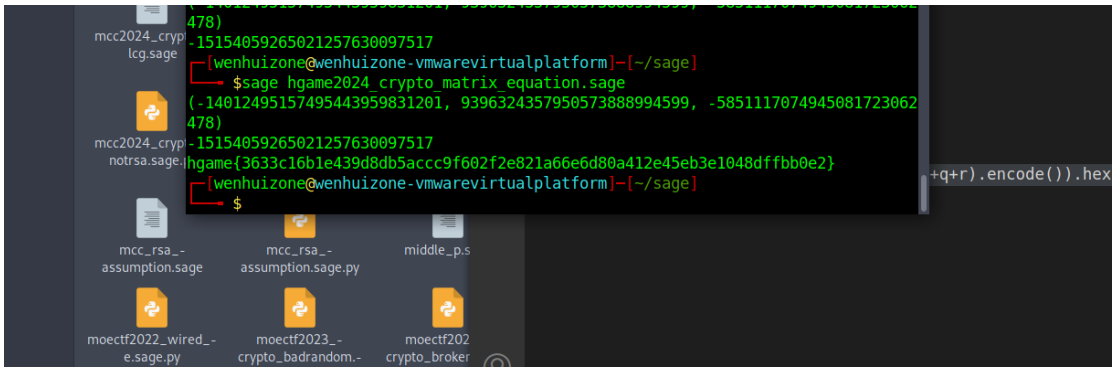
格出来的数据发现是 83 位，即可求得 temp，然后通过 pq 来求解 r 即可

Exp：

```
#from sage.all import *
from Crypto.Util.number import *
from gmpy2 import *
from tqdm import *
import hashlib


k1 =
737153298772153401459512383432471562821657053960747864832566998176512555
709671
k2 =
613619706622698697382703285238977654084439071983136324100684542237178240
```

276837

```
M = Matrix(ZZ,3,3,[[1,0,2^256],
                   [0,1,k1],
                   [0,0,k2]
                   ])
ll = M.LLL()[0]
print(ll)
p = abs(ll[0])
q = -ll[1]
temp = abs(ll[2])
r = (temp -p*2^256-k1*q)/k2
print(r)

flag='hgame{'+hashlib.sha256(str(p+q+r).encode()).hexdigest()+'}'
print(flag)
```



Flag：
hgame{3633c16b1e439d8db5accc9f602f2e821a66e6d80a412e45eb3e1048dffbb0e2}

Crypto: [hgame2024-week3] HNP（LLL,格基规约，格配平）

Code:

```
from Crypto.Util.number import *
from secret import flag

def encrypt(m,p,t):
    return [(ti*m)%p for ti in t]

m=bytes_to_long(flag[:63])
length=m.bit_length()+8
p=getStrongPrime(length)
n=32
t=[getRandomRange(0,p) for _ in range(n)]
```

```
enc=encrypt(m,p,t)
res=[i%(2**n+1) for i in enc]

print(f'p={p}')
print(f't={t}')
print(f'res={res}')

"""
p=113062992417749500532695471032846374144078351257772452040693675676910219288647732075487310515928535152062323659011697780480841465208290323393283263913558053
t=[33220085552551293368213097014829969330453797924325322515795645812110726774032449704233579122984444574573066598012001881665691325606590083569527405993716888,827676426026485881184521157841502334394263461352208863102119943306692429104985860704596069057403576137039426315498135172849430973790112170328882261636726 6,98722917369229744564204184636011290942272319792183859851496611327924676219407225807453278354053748262937913328151764587505489427570240173828815172849916 46,40215217451425358131536699611464574066407919358447960053440738862896684648850114158877557879039278247628331581306150183266661183831286275356236390468177 99,24569151076141700493541155834378165089870615699969211988778938492838766214386066952596557490584021813819164202001474086538804476667616708172536787956586,32185011565208485728614588311238226897020352425148035050491017799962317508750363445643226000868613614146092012148222629084280910973827817708509290674042 10,35634059873983750763276334440364921630049587148286858462028186103204393063969124254203910701170698755837868193231733429511725940466520172975528135015571 59,49147090456938630385982251245345150489933107702861050707255136674359837898475472251800248243214587612623908174878616755954665135389013734221492361339263 54,10800566112999947911006702454427389510409658644419749067440812458744391509925306994806187389406032718319773665587324010542068486131582672363925769248595266,62336492005220979079812873108919481313890969103913793527503733950362212632592877303750125472285168431802401410814952521508326573371280916234455399842732 4,49184210976284306138012655258705610412300110298188512910868629705086215290744976016787749212859127455898405104596775220748875761520153569845925896498444 31,
```

744573335721584737007069613665368974871802808036481226394778574735325893696897818347154970616636424314897215421505522485791883493770755505324618482209560 2,
933353475504922562753028424938843869400260264504793386545315983679666719896605817798850018407345438618408093472753720057545759897612166737380144139593244 0,
501085480317997044583879157532112791127831163523007663902341157114848890340061012124861730777387261274322899889298620020271349657037544725525863093215882 2,
600064506846256981964846107014055752114480101349010663235683632500254640087146395722858114395459100539853325221842997048611549053558407178626081877316632 4,
800726090912466938186203490155611124578050598708299080438081479720032222894243267393994469306247017825686736660233161236317640835630464167245945651797856 0,
101797391753738833769295320263891357921292337306012786875070414294389455985239957001846223596606059109328031417855987583262548864484810463076660428358297 25,
839007276771739570192628977943305567286388033603183700911910344867523236294222363312932830911815827383596156743659123492278395337331976783587726684954529 2,
787501191156296787467611368069392923028386684147564116285466529311134446770942440862319837094279709996462544751279713819285300912688885328352603441100751 3,
529377281102001250102012477521477019323465521031934305864867541111521045368075307004282183508261963434150068089232300211895355774611691809366176946464206 8,
261379727942677454030646193131919365799989212984483215965877171738712024679568967823127537149955652239606159188243142631084197471341997404588302161398770 5,
965812601213321780412663000523607351348521539081297797466002905352266528255096504028825607494524685074469451954335877725292966156163624116157593706152171 1,
298253522084497762177513940635752887601934938563481179548023067798234569718358620366909499803999568397393972164488754390749496382496804219935394512036750 5,
107289984878191849357180490850397539311037762262082755398160292401340078782643246498566039415279868796667596686125847400130898160017838981308638814854641,
120993130590874228473811314869823704699012435303134640953201808807618070048912918046616664677916248813062043597607873728870402493717351447905456920806865,
225304065277179628426625426171980576810274065309744632586978381220117114415076887588596372932491571481271913824778419475263692826771234473619800 0

611708630089,
865000727215428305735066431150588753584126876742454501690141898955562086909114565121644872320024091414388277461667896872552391431096535687568120729524243 4,
962874782910758465001415607992810880168715802908622173088399974904453284648966611547399300544219285917193188279597377413130990002128731905921610593967075 7,
108469369515220937060920279081316799124326897124519207184390967064355339269962157661919670526679660659170066915657716957727987112028121807829012502496130 72,
160686565122798873666412702167868929998904543999833660356223290886340577847452091517076677181133631965579274659098174061782356481357311841006497608198923 7,
623906365759172109773504940961087294121407869933013682659295854921248180297397310437454855518490792925503157052534300751843435769048042998101678111024961 2,
185536591638711462058102993970705370106247674523557868355806379660474444805027813895435950692287596753756735957566239429757995837210748427636092056773045 8]
res=[2150646508, 1512876052, 2420557546, 2504482055, 892924885, 213721693, 2708081441, 1242578136, 717552493, 3210536920, 2868728798, 1873446451, 645647556, 2863150833, 2481560171, 2518043272, 3183116112, 3032464437, 934713925, 470165267, 1104983992, 194502564, 1621769687, 3844589346, 21450588, 2520267465, 2516176644, 3290591307, 3605562914, 140915309, 3690380156, 3646976628]
"""

根据题意，加密线性关系为：
A*m+l*p-(2**32+1)*k-B=0
可以构造矩阵，一般构造方阵：

$$L=\begin{pmatrix} p & & & & & & & & & \\ & p & & & & & & & & \\ & & \ddots & & & & & & & \\ & & & p & & & & & & \\ 2^{32}+1 & & & & 1 & & & & & \\ & 2^{32}+1 & & & & 1 & & & & \\ & & \ddots & & & & \ddots & & & \\ & & & 2^{32}+1 & & & & 1 & & \\ A_0 & A_1 & \cdots & A_{31} & & & & & 1 & \\ B_0 & B_1 & \cdots & B_{31} & & & & & & 2^{508} \end{pmatrix}_{66*66}$$

$$\begin{pmatrix} l_0 & \cdots & l_{31} & k_0 & k_1 & \cdots & k_{31} & m & -1 \end{pmatrix} L=\begin{pmatrix} 0 & \cdots & 0 & k_0 & k_1 & \cdots & k_{31} & m & 2^{508} \end{pmatrix}$$

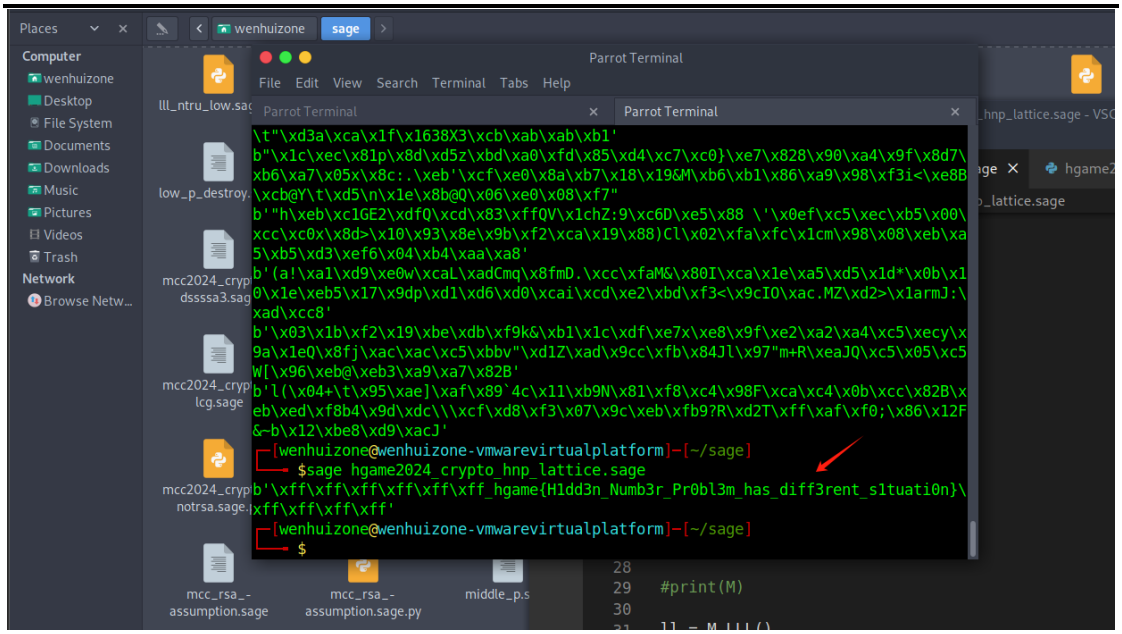
主要要学习下配平

目标是对于输出向量输出的值都差不多，0 除外，k 的比特位数为 p 的 bit 位数 512-32=480 位，就是与 m 504 位相差 24 位，因此在 k 处需要乘 2**24 配平，同事对于 0 项，采用大系数配平即可

Exp：

```
from sage.all import *
from Crypto.Util.number import *
from gmpy2 import *

p = 11306299241774950053269547103284637414407835125777245204069367567691021928864773207548731051592853515206232365901169778048084146520829032339328263913558053
A = […]
B = [2150646508, 1512876052, 2420557546, 2504482055, 892924885, 213721693, 2708081441, 1242578136, 717552493, 3210536920, 2868728798, 1873446451, 645647556, 2863150833, 2481560171, 2518043272, 3183116112, 3032464437, 934713925, 470165267, 1104983992, 194502564, 1621769687, 3844589346, 21450588, 2520267465, 2516176644, 3290591307, 3605562914, 140915309, 3690380156, 3646976628]
n = 32

M = Matrix(ZZ,66,66)
T = 2^1000

for i in range(0,32):
    M[i,i] = p*T
    M[64,i] = A[i]*T
    M[65,i] = B[i]*T
    M[32+i,32+i] = 1*(2^32)
    M[32+i,i] = (2^32+1)*T
    #M[i,32] = M[i,33] = 0
    M[i,64] = M[i,65] = 0

M[64,64] = 1
M[64,65]= 0
M[65,64] = 0
M[65,65] = 2^504

#print(M)

ll = M.LLL()
#print(ll)
for i in ll:
    flag = long_to_bytes(abs(int(i[-2])))
```

print(flag)



Flag：hgame{H1dd3n_Numb3r_Pr0bl3m_has_diff3rent_s1tuati0n}

RE: [hgame2024-week3] findme（花指令，变种 RC4）

进去主函数，都是一些假的 flag，不过发现 buf 处存在 MZ，一看就是 win 的 exe 头，将其 dump 出来，获得一个新的程序，

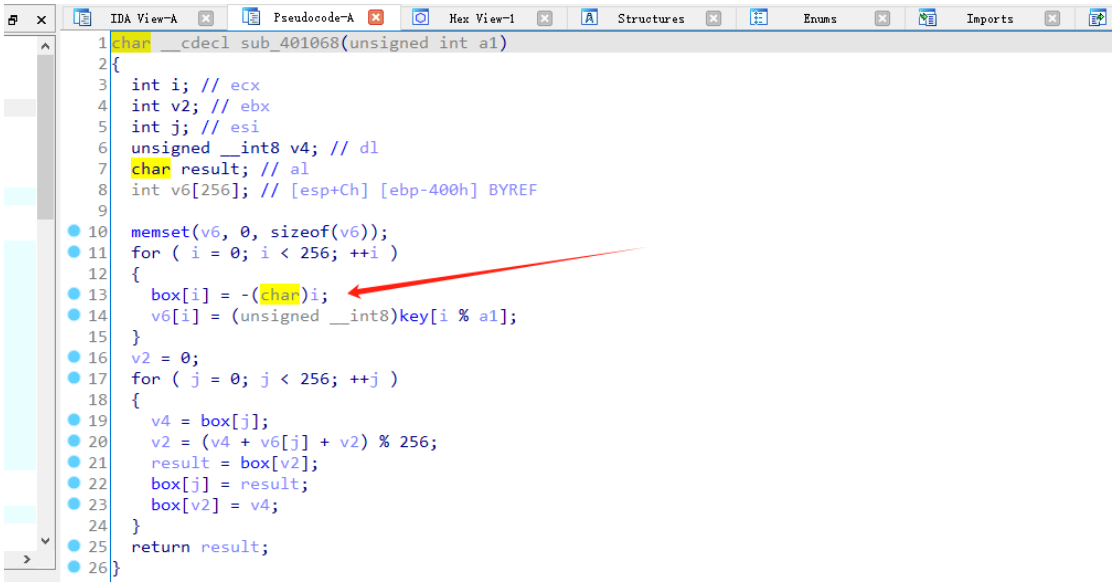
```
>>> chr(0x5a)
'Z'
>>> lst = [0x0000004D, 0x0000005A, 0x00000090, 0x00000000, 0x00000003, 0x00000000, 0x00000000,
>>> f = open('d:/hgame','wb')
>>> strr = ''
>>> for i in lst:
...     strr += chr(i)
... f.write(strr)
>>> f.close()
>>> 'a'*32
```

丢进 IDA 进行逆向

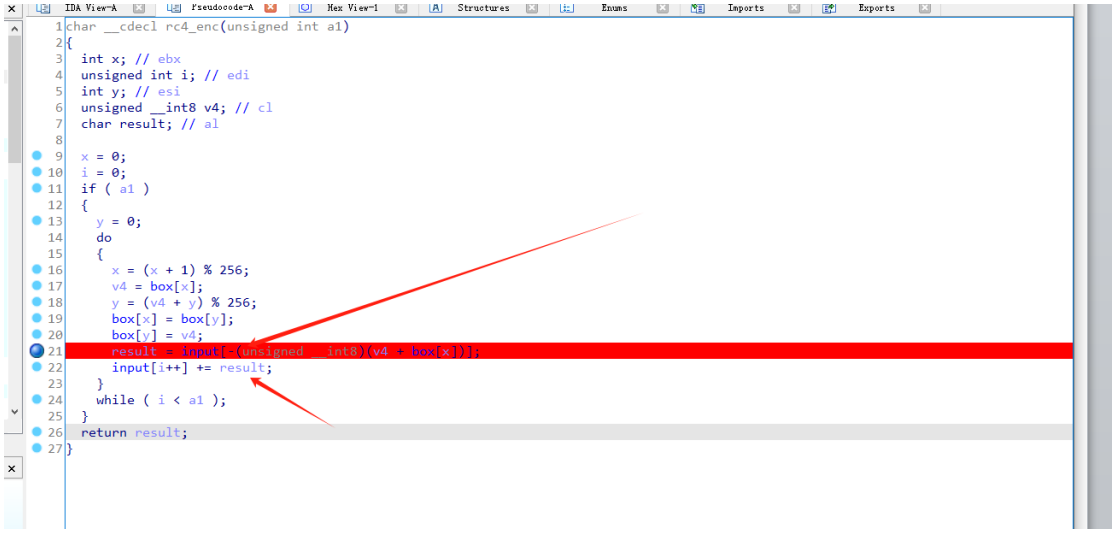
```
.text:00401190 _main:                                  ; CODE XREF: __scrt_common_main_seh(void)+F5↓p
.text:00401190                 push    ebp
.text:00401191                 mov     ebp, esp
.text:00401193                 push    ecx
.text:00401194                 push    ebx
.text:00401195                 push    esi
.text:00401196                 push    edi
.text:00401197                 jz      short loc_40119C
.text:00401199                 jnz     short loc_40119C
.text:00401199 ; ---------------------------------------------------------------------------
.text:0040119B                 db 0C7h
.text:0040119C ; ---------------------------------------------------------------------------
.text:0040119C
.text:0040119C loc_40119C:                             ; CODE XREF: .text:00401197↑j
.text:0040119C                                         ; .text:00401199↑j
.text:0040119C                 push    offset aPlzInputFlag ; "plz input flag:\n"
.text:004011A1                 call    sub_40100C
.text:004011A6                 mov     dword ptr [esp], offset byte_403490
.text:004011AD                 push    offset a32s     ; "%32s"
.text:004011B2                 call    sub_40103A
.text:004011B7                 pop     ecx
.text:004011B8                 pop     ecx
.text:004011B9                 mov     ecx, offset aDeadbeef ; "deadbeef"
.text:004011BE                 lea     edx, [ecx+1]
.text:004011C1
.text:004011C1 loc_4011C1:                             ; CODE XREF: .text:004011C6↓j
.text:004011C1                 mov     al, [ecx]
.text:004011C3                 inc     ecx
.text:004011C4                 test    al, al
.text:004011C6                 jnz     short loc_4011C1
.text:004011C8                 sub     ecx, edx
.text:004011CA                 mov     [ebp-4], ecx
.text:004011CD                 jz      short loc_4011D2
.text:004011CF                 jnz     short loc_4011D2
.text:004011CF ; ---------------------------------------------------------------------------
.text:004011D1                 db 0C7h
.text:004011D2 ; ---------------------------------------------------------------------------
.text:004011D2
.text:004011D2 loc_4011D2:                             ; CODE XREF: .text:004011CD↑j
.text:004011D2                                         ; .text:004011CF↑j
000005B2 004011B2: .text:004011B2 (Synchronized with Hex View-1)
```

发现很多 jz，jnz 都直接跳的，那就是典型的花指令，改 jmp，总共 11 处，去除花指令，恢复完整代码

点进去看，以后发现一个魔改的 rc4 的算法，其中 key=deadbeef，但是有两处变化，1 个是初始的 s 盒发生了变化，动态调一下可以发现规律，另一个是原版 RC4 中的异或无了，改为了加法，另外索引值也有些魔改



```
1  char __cdecl sub_401068(unsigned int a1)
2  {
3    int i; // ecx
4    int v2; // ebx
5    int j; // esi
6    unsigned __int8 v4; // dl
7    char result; // al
8    int v6[256]; // [esp+Ch] [ebp-400h] BYREF
9
10   memset(v6, 0, sizeof(v6));
11   for ( i = 0; i < 256; ++i )
12   {
13     box[i] = -(char)i;
14     v6[i] = (unsigned __int8)key[i % a1];
15   }
16   v2 = 0;
17   for ( j = 0; j < 256; ++j )
18   {
19     v4 = box[j];
20     v2 = (v4 + v6[j] + v2) % 256;
21     result = box[v2];
22     box[j] = result;
23     box[v2] = v4;
24   }
25   return result;
26 }
```

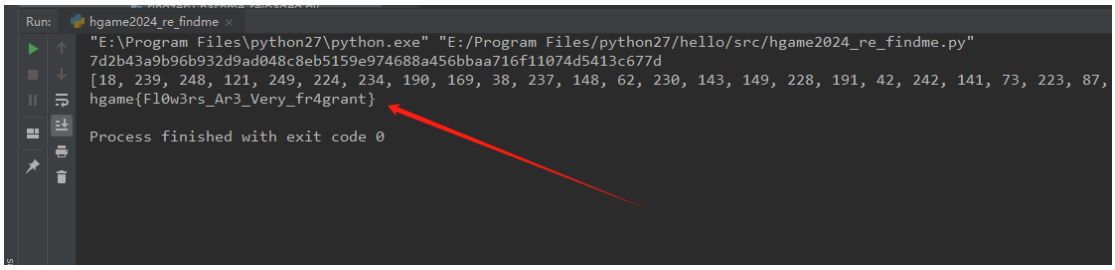


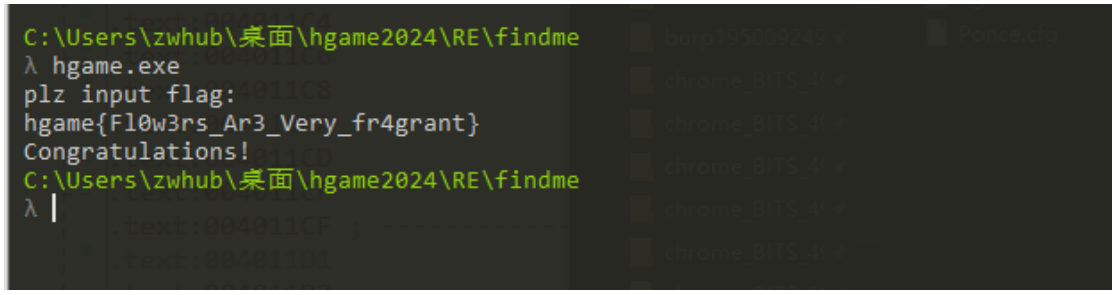
```
1  char __cdecl rc4_enc(unsigned int a1)
2  {
3    int x; // ebx
4    unsigned int i; // edi
5    int y; // esi
6    unsigned __int8 v4; // cl
7    char result; // al
8
9    x = 0;
10   i = 0;
11   if ( a1 )
12   {
13     y = 0;
14     do
15     {
16       x = (x + 1) % 256;
17       v4 = box[x];
18       y = (v4 + y) % 256;
19       box[x] = box[y];
20       box[y] = v4;
21       result = input[-(unsigned __int8)(v4 + box[x])];
22       input[i++] += result;
23     }
24     while ( i < a1 );
25   }
26   return result;
27 }
```

编写 exp：

```
def decrypt(data, key):
    """RC4 algorithm"""
    x = 0
    box = [0, 255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245,
244, 243, 242, 241, 240, 239, 238, 237, 236, 235, 234, 233, 232,
231, 230, 229, 228, 227, 226, 225, 224, 223, 222, 221, 220, 219,
218, 217, 216, 215, 214, 213, 212, 211, 210, 209, 208, 207, 206,
205, 204, 203, 202, 201, 200, 199, 198, 197, 196, 195, 194, 193,
192, 191, 190, 189, 188, 187, 186, 185, 184, 183, 182, 181, 180,
179, 178, 177, 176, 175, 174, 173, 172, 171, 170, 169, 168, 167,
166, 165, 164, 163, 162, 161, 160, 159, 158, 157, 156, 155, 154,
153, 152, 151, 150, 149, 148, 147, 146, 145, 144, 143, 142, 141,
140, 139, 138, 137, 136, 135, 134, 133, 132, 131, 130, 129, 128,
127, 126, 125, 124, 123, 122, 121, 120, 119, 118, 117, 116, 115,
114, 113, 112, 111, 110, 109, 108, 107, 106, 105, 104, 103, 102,
101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86,
```

```
85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70,
69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54,
53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38,
37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22,
21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4,
3, 2, 1]
    for i in range(256):
        x = (x + box[i] + ord(key[(i % len(key))])) % 256
        box[i], box[x] = box[x], box[i]

    print(box)
    x = y = 0
    #y = x
    #x = 0
    #y = box[x]
    out = []
    for char in data:
        x = (x + 1) % 256
        y = (y + box[x]) % 256
        box[x], box[y] = box[y], box[x]
        out.append(chr((ord(char) - box[(-(box[x] + box[y]) %
256)])%128))

    return ('').join(out)



lst = [0x7D, 0x2B, 0x43, 0xA9, 0xB9, 0x6B, 0x93, 0x2D, 0x9A, 0xD0,
0x48, 0xC8, 0xEB, 0x51, 0x59, 0xE9, 0x74, 0x68, 0x8A, 0x45, 0x6B,
0xBA, 0xA7, 0x16, 0xF1, 0x10, 0x74, 0xD5, 0x41, 0x3C, 0x67, 0x7D]

cipher = ''

for i in range(len(lst)):
    cipher += chr(lst[i])
print cipher.encode('hex')
print decrypt(cipher,'deadbeef')
```

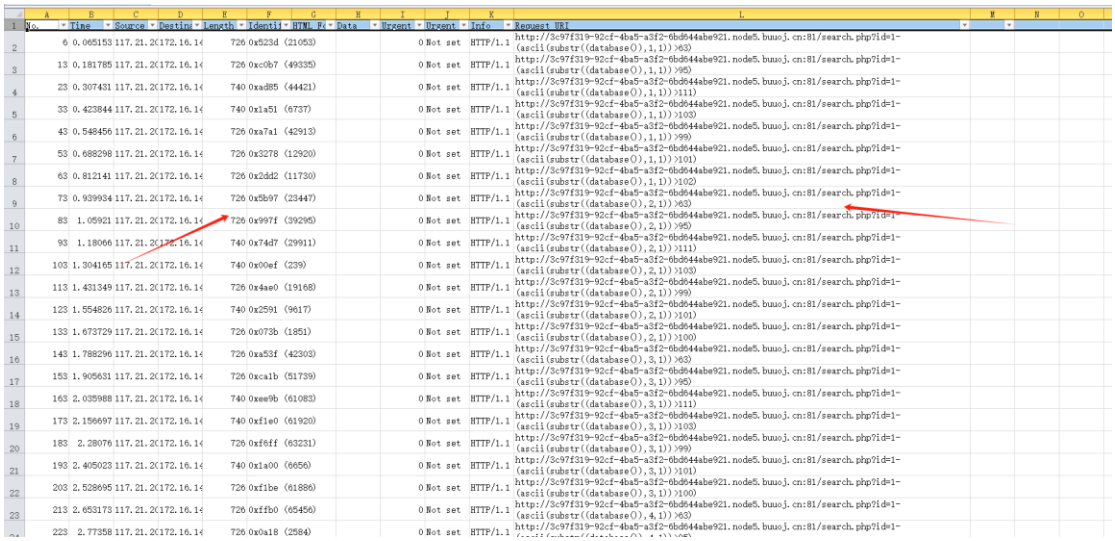




Flag：hgame{Fl0w3rs_Ar3_Very_fr4grant}

Misc: [hgame2024-week3] Blind SQL Injection（流量分析）

打开包很明显是一个注入过程，筛选出源地址为攻击地址且为 http 协议的导出一份 csv 进行分析



No.	Time	Source	Destination	Length	Identification	HTML Form URL Encoded	Data	Urgent	Urgent	Info	Request URI
6	0.065153	117.21.2(172.16.14		726	0x523d (21053)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>63)
13	0.181785	117.21.2(172.16.14		726	0xc0b7 (49335)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>95)
23	0.307431	117.21.2(172.16.14		740	0xad85 (44421)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>111)
33	0.423844	117.21.2(172.16.14		740	0x1a51 (6737)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>103)
43	0.548456	117.21.2(172.16.14		726	0xa7a1 (42913)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>99)
53	0.688298	117.21.2(172.16.14		726	0x3278 (12920)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>101)
63	0.812141	117.21.2(172.16.14		726	0x2dd2 (11730)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),1,1))>102)
73	0.939934	117.21.2(172.16.14		726	0x5b97 (23447)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>63)
83	1.05921	117.21.2(172.16.14		726	0x997f (39295)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>95)
93	1.18066	117.21.2(172.16.14		740	0x74d7 (29911)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>111)
103	1.304165	117.21.2(172.16.14		740	0x00ef (239)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>103)
113	1.431349	117.21.2(172.16.14		726	0x4ae0 (19168)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>99)
123	1.554826	117.21.2(172.16.14		740	0x2591 (9617)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>101)
133	1.673729	117.21.2(172.16.14		726	0x073b (1851)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),2,1))>100)
143	1.788296	117.21.2(172.16.14		726	0xa53f (42303)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>63)
153	1.905631	117.21.2(172.16.14		726	0xca1b (51739)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>95)
163	2.035988	117.21.2(172.16.14		740	0xee9b (61083)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>111)
173	2.156697	117.21.2(172.16.14		740	0xf1e0 (61920)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>103)
183	2.28076	117.21.2(172.16.14		726	0xf6ff (63231)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>99)
193	2.405023	117.21.2(172.16.14		740	0x1a00 (6656)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>101)
203	2.528695	117.21.2(172.16.14		726	0xf1be (61886)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),3,1))>100)
213	2.653173	117.21.2(172.16.14		726	0xffb0 (65456)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~(ascii(substr((database()),4,1))>63)
223	2.77358	117.21.2(172.16.14		726	0x0a18 (2584)			0	Not set	HTTP/1.1	http://3c97f319-92cf-4ba5-a3f2-6bd644abe921.node5.buuoj.cn:81/search.php?id=1~

长度为 740 的为正确响应，726 的为非正确响应，那么只要挑选出每次注入的最后一次 740 的数值即为注入结果

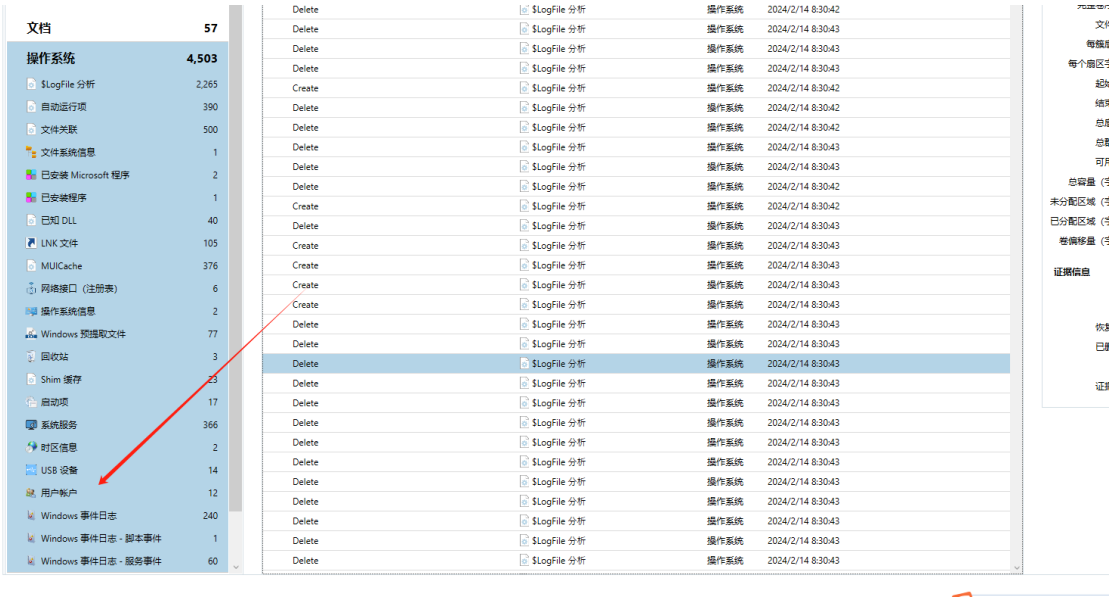
同时，注意到，此此次注入使用了 reverse 函数，注入结果是反序的，只要反下即为 flag

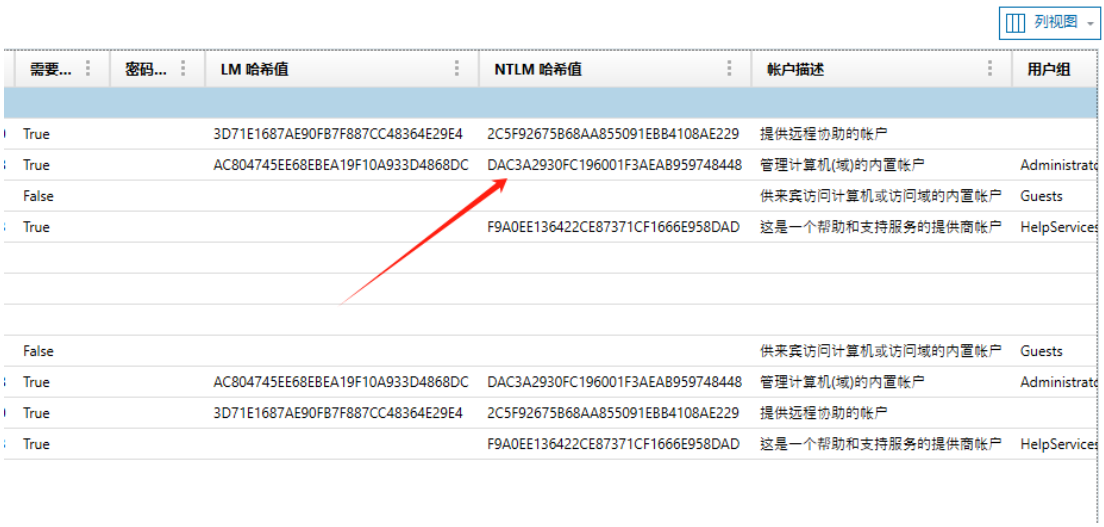
```
PyDev console: starting.

Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)]
>>> lst = [125, 102, 50, 102, 97, 56, 50, 57, 53, 99, 56, 51, 100, 45, 54, 99, 97, 98,
>>> flag = ''
>>> for i in lst:
...     flag += chr(i)
... print flag
}f2fa8295c83d-6cab-89e4-5271-7efababc{galf
>>> print flag[::-1]
flag{cbabafe7-1725-4e98-bac6-d38c5928af2f}

>>>
```

Flag：flag{cbabafe7-1725-4e98-bac6-d38c5928af2f}

Misc: [hgame2024-week3] 简单 vmdk 取证+veracrypt (取证)

需要分析 vmdk 中用户的 NTLM hash 和密码，同时另一个题问了管理和这个有关系，那么就多翻翻了，先说第一个
第一个主要找到 sam 表，用取证工具直接搞了一下



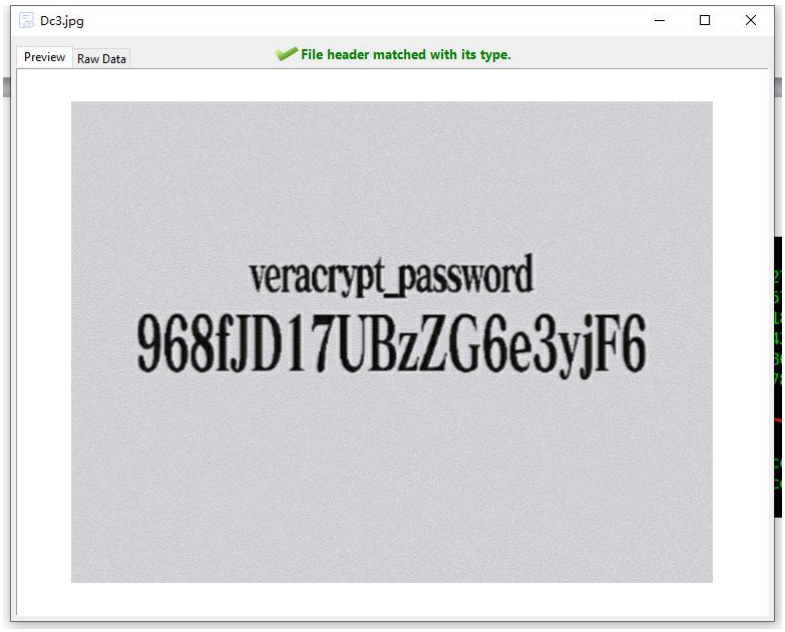


需要...	密码...	LM 哈希值	NTLM 哈希值	帐户描述	用户组
True		3D71E1687AE90FB7F887CC48364E29E4	2C5F92675B68AA855091EBB4108AE229	提供远程协助的帐户	
True		AC804745EE68EBEA19F10A933D4868DC	DAC3A2930FC196001F3AEAB959748448	管理计算机(域)的内置帐户	Administrato
False				供来宾访问计算机或访问域的内置帐户	Guests
True			F9A0EE136422CE87371CF1666E958DAD	这是一个帮助和支持服务的提供商帐户	HelpServices
False				供来宾访问计算机或访问域的内置帐户	Guests
True		AC804745EE68EBEA19F10A933D4868DC	DAC3A2930FC196001F3AEAB959748448	管理计算机(域)的内置帐户	Administrato
True		3D71E1687AE90FB7F887CC48364E29E4	2C5F92675B68AA855091EBB4108AE229	提供远程协助的帐户	
True			F9A0EE136422CE87371CF1666E958DAD	这是一个帮助和支持服务的提供商帐户	HelpServices

Ntlmhash 为：DAC3A2930FC196001F3AEAB959748448

Flag1：hgame{DAC3A2930FC196001F3AEAB959748448_Admin1234}

Diskgenius 加载 vmdk，发现一张删除的图片里有个 veracrypt 的密码，直接加载即可获得 flag

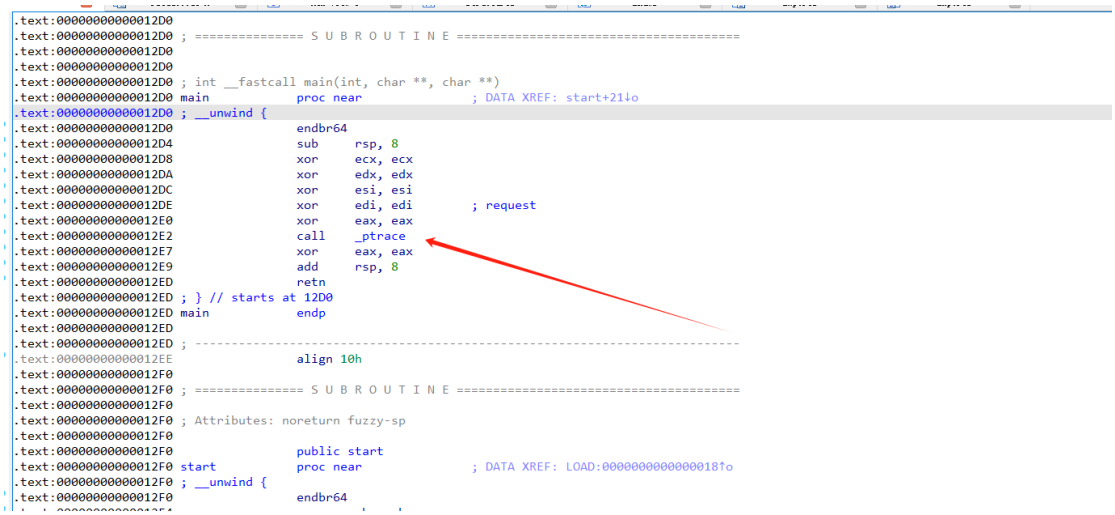


Flag2: hgame{happy_new_year_her3_1s_a_redbag_key_41342177}

RE: [hgame2024-week3] findme（反调试，变种 RC4）

本题考察了反调试和变种的 RC4

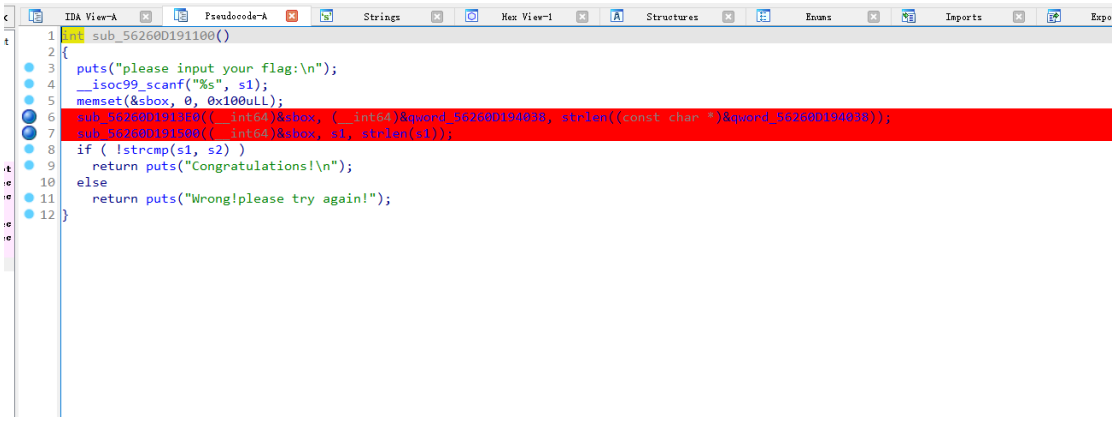
进去主函数，发现 ptrace 反调试，全部 nop 掉



```
.text:00000000000012D0
.text:00000000000012D0 ; =============== S U B R O U T I N E =======================================
.text:00000000000012D0
.text:00000000000012D0
.text:00000000000012D0 ; int __fastcall main(int, char **, char **)
.text:00000000000012D0 main            proc near               ; DATA XREF: start+21↓o
.text:00000000000012D0 ; __unwind {
.text:00000000000012D0                 endbr64
.text:00000000000012D4                 sub     rsp, 8
.text:00000000000012D8                 xor     ecx, ecx
.text:00000000000012DA                 xor     edx, edx
.text:00000000000012DC                 xor     esi, esi
.text:00000000000012DE                 xor     edi, edi        ; request
.text:00000000000012E0                 xor     eax, eax
.text:00000000000012E2                 call    _ptrace
.text:00000000000012E7                 xor     eax, eax
.text:00000000000012E9                 add     rsp, 8
.text:00000000000012ED                 retn
.text:00000000000012ED ; } // starts at 12D0
.text:00000000000012ED main            endp
.text:00000000000012ED
.text:00000000000012ED ; ---------------------------------------------------------------------------
.text:00000000000012EE                 align 10h
.text:00000000000012F0
.text:00000000000012F0 ; =============== S U B R O U T I N E =======================================
.text:00000000000012F0
.text:00000000000012F0 ; Attributes: noreturn fuzzy-sp
.text:00000000000012F0
.text:00000000000012F0                 public start
.text:00000000000012F0 start           proc near               ; DATA XREF: LOAD:0000000000000018↑o
.text:00000000000012F0 ; __unwind {
.text:00000000000012F0                 endbr64
```

```
.text:000056260D1912D0 ; __int64 __fastcall sub_56260D1912D0(int, char **, char **)
.text:000056260D1912D0 sub_56260D1912D0 proc near              ; DATA XREF: start+21↓o
.text:000056260D1912D0 ; __unwind { // 56260D190000
.text:000056260D1912D0                 endbr64
.text:000056260D1912D4                 sub     rsp, 8
.text:000056260D1912D8                 nop
.text:000056260D1912D9                 nop
.text:000056260D1912DA                 nop
.text:000056260D1912DB                 nop
.text:000056260D1912DC                 nop
.text:000056260D1912DD                 nop
.text:000056260D1912DE                 nop
.text:000056260D1912DF                 nop
.text:000056260D1912E0                 nop
.text:000056260D1912E1                 nop
.text:000056260D1912E2                 nop
.text:000056260D1912E3                 nop
.text:000056260D1912E4                 nop
.text:000056260D1912E5                 nop
.text:000056260D1912E6                 nop
.text:000056260D1912E7                 xor     eax, eax
.text:000056260D1912E9                 add     rsp, 8
.text:000056260D1912ED                 retn
.text:000056260D1912ED ; } // starts at 56260D1912D0
.text:000056260D1912ED sub_56260D1912D0 endp
.text:000056260D1912ED
.text:000056260D1912ED ; ---------------------------------------------------------------------------
.text:000056260D1912EE                 align 10h
.text:000056260D1912F0
```

跟踪一下发现函数主要的逻辑的部分



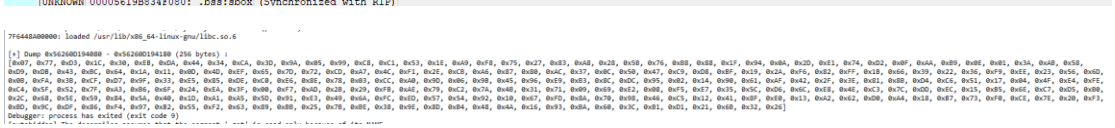

```
int sub_56260D191100()
{
  puts("please input your flag:\n");
  __isoc99_scanf("%s", s1);
  memset(&sbox, 0, 0x100uLL);
  sub_56260D1913E0((__int64)&sbox, (__int64)&qword_56260D194038, strlen((const char *)&qword_56260D194038));
  sub_56260D191500((__int64)&sbox, s1, strlen(s1));
  if ( !strcmp(s1, s2) )
    return puts("Congratulations!\n");
  else
    return puts("Wrong!please try again!");
}
```

```
unsigned __int64 __fastcall sub_56260D1913E0(__int64 a1, __int64 a2, unsigned __int64 a3)
{
  unsigned __int64 i; // rcx
  __int64 v4; // rcx
  int v5; // eax
  unsigned __int8 v6; // si
  unsigned int v7; // edx
  unsigned __int8 *v8; // rdx
  _DWORD v10[258]; // [rsp+0h] [rbp-418h] BYREF
  unsigned __int64 v11; // [rsp+408h] [rbp-10h]

  v11 = __readfsqword(0x28u);
  memset(v10, 0, 0x400uLL);
  for ( i = 0LL; i != 256; ++i )
  {
    *(_BYTE *)(a1 + i) = i;
    v10[i] = *(unsigned __int8 *)(a2 + i % a3);
  }
  v4 = 0LL;
  v5 = 0;
  do
  {
    v6 = *(_BYTE *)(a1 + v4);
    v7 = (v10[v4] + v6 + v5) >> 31;
    v5 = (unsigned __int8)(HIBYTE(v7) + LOBYTE(v10[v4]) + v6 + v5) - HIBYTE(v7);
    v8 = (unsigned __int8 *)(a1 + v5);
    *(_BYTE *)(a1 + v4++) = *v8;
    *v8 = v6;
  }
  while ( v4 != 256 );
  return __readfsqword(0x28u) ^ v11;
}
```

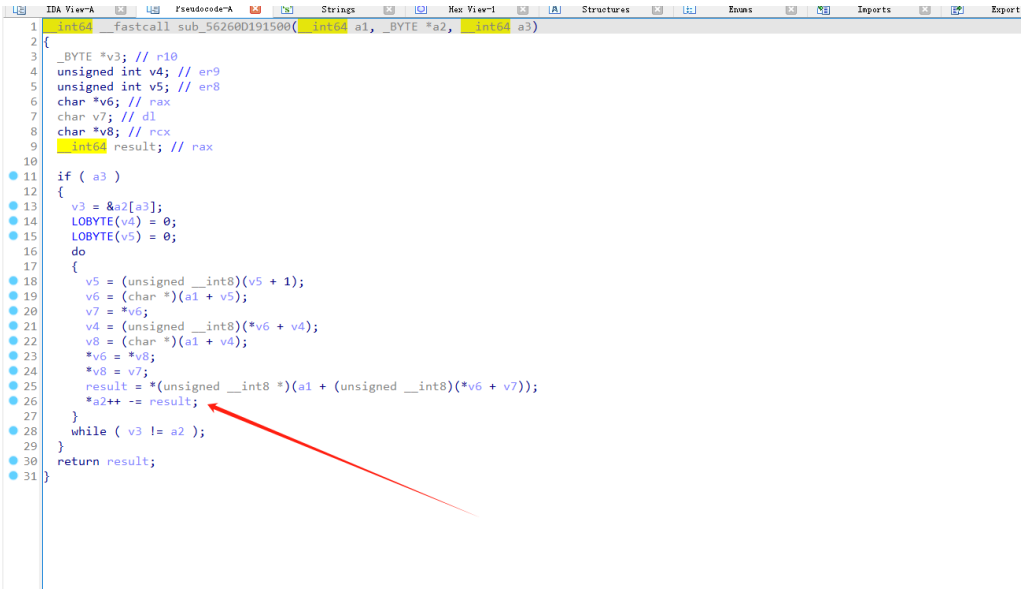
疑似 rc4 算法

猜测 256 的长度的数据是 S 盒，动调发现 key 为 keykey，暂且记下，第一个函数是初始化 s 盒，通过动调可以获取到 s 盒

```
.bss:00005619B834F061 align 20h
.bss:00005619B834F080 sbox db    7                  ; DATA XREF: sub_5619B834C100+29↑o
.bss:00005619B834F080                               ; sub_5619B834C100+78↑o ...
.bss:00005619B834F081 db  77h ; w
.bss:00005619B834F082 db 0D3h
.bss:00005619B834F083 db  1Ch
.bss:00005619B834F084 db  30h ; 0
.bss:00005619B834F085 db 0EBh
.bss:00005619B834F086 db 0DAh
.bss:00005619B834F087 db  44h ; D
.bss:00005619B834F088 db  34h ; 4
.bss:00005619B834F089 db 0CAh
.bss:00005619B834F08A db  3Dh ; =
.bss:00005619B834F08B db  9Ah
.bss:00005619B834F08C db    5
.bss:00005619B834F08D db  99h
.bss:00005619B834F08E db 0C8h
.bss:00005619B834F08F db 0C1h
.bss:00005619B834F090 db  53h ; S
.bss:00005619B834F091 db  1Eh
.bss:00005619B834F092 db 0A9h
.bss:00005619B834F093 db 0F8h
.bss:00005619B834F094 db  75h ; u

UNKNOWN 00005619B834F080: .bss:sbox (Synchronized with RIP)
```



第二步是加密



```
__int64 __fastcall sub_56260D191500(__int64 a1, _BYTE *a2, __int64 a3)
{
  _BYTE *v3; // r10
  unsigned int v4; // er9
  unsigned int v5; // er8
  char *v6; // rax
  char v7; // dl
  char *v8; // rcx
  __int64 result; // rax

  if ( a3 )
  {
    v3 = &a2[a3];
    LOBYTE(v4) = 0;
    LOBYTE(v5) = 0;
    do
    {
      v5 = (unsigned __int8)(v5 + 1);
      v6 = (char *)(a1 + v5);
      v7 = *v6;
      v4 = (unsigned __int8)(*v6 + v4);
      v8 = (char *)(a1 + v4);
      *v6 = *v8;
      *v8 = v7;
      result = *(unsigned __int8 *)(a1 + (unsigned __int8)(*v6 + v7));
      *a2++ -= result;
    }
    while ( v3 != a2 );
  }
  return result;
}
```

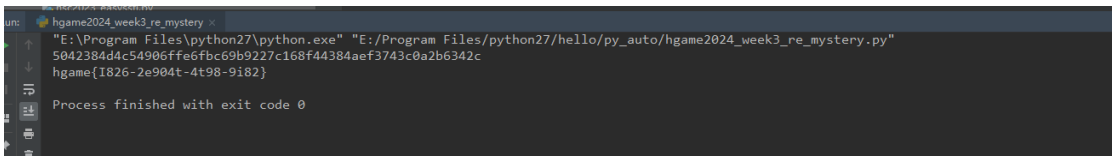
最后一步是用减法代替了原版 rc4 的减法

于是和上一题 findme 差不多，直接写 exp 即可

Exp：

```
def decrypt(data):
    """RC4 algorithm"""
    box = [0x07, 0x77, 0xD3, 0x1C, 0x30, 0xEB, 0xDA, 0x44, 0x34,
0xCA, 0x3D, 0x9A, 0x05, 0x99, 0xC8, 0xC1, 0x53, 0x1E, 0xA9, 0xF8,
0x75, 0x27, 0x83, 0xA8, 0x28, 0x5B, 0x76, 0xB8, 0x88, 0x1F, 0x94,
0x0A, 0x2D, 0xE1, 0x74, 0xD2, 0x0F, 0xAA, 0xB9, 0x0E, 0x01, 0x3A,
0xAB, 0x58, 0xD9, 0xDB, 0x43, 0xBC, 0x64, 0x1A, 0x11, 0x0D, 0x4D,
0xEF, 0x65, 0x7D, 0x72, 0xCD, 0xA7, 0x4C, 0xF1, 0x2E, 0xCB, 0xA6,
0x87, 0x80, 0xAC, 0x37, 0x0C, 0x50, 0x47, 0xC9, 0xD8, 0xBF, 0x19,
0x2A, 0xF6, 0x82, 0xFF, 0x1B, 0x66, 0x39, 0x22, 0x36, 0xF9, 0xEE,
0x23, 0x56, 0x6D, 0x0B, 0xFA, 0x3B, 0xCF, 0xD7, 0x9F, 0x33, 0xE5,
0x85, 0xDE, 0xC0, 0xE6, 0x8E, 0x78, 0x03, 0xCC, 0xA0, 0x9D, 0x06,
0x9B, 0x45, 0x96, 0xE9, 0xB3, 0x8C, 0xDC, 0x95, 0x02, 0x14, 0x90,
0x61, 0xAF, 0x42, 0x2F, 0x3E, 0x81, 0x8B, 0xD4, 0xC6, 0x51, 0x17,
0x04, 0x4F, 0xE4, 0xFE, 0xC4, 0x5F, 0x52, 0x7F, 0xA3, 0xB6, 0x6F,
0x24, 0xEA, 0x3F, 0x00, 0xF7, 0xAD, 0x2B, 0x29, 0xFB, 0xAE, 0x79,
0xC2, 0x7A, 0x4B, 0x31, 0x71, 0x09, 0x69, 0xE2, 0x08, 0xF5, 0xE7,
0x35, 0x5C, 0xD6, 0x6C, 0xE8, 0x4E, 0xC3, 0x7C, 0xDD, 0xEC, 0x15,
0xB5, 0x6E, 0xC7, 0xD5, 0xB0, 0x2C, 0x68, 0x5E, 0x59, 0x84, 0x5A,
0x40, 0x1D, 0xA1, 0xA5, 0x5D, 0x91, 0xE3, 0x49, 0x6A, 0xFC, 0xED,
0x57, 0x54, 0x92, 0x10, 0x67, 0xFD, 0x8A, 0x70, 0x98, 0x46, 0xC5,
0x12, 0x41, 0x8F, 0xE0, 0x13, 0xA2, 0x62, 0xD0, 0xA4, 0x18, 0xB7,
0x73, 0xF0, 0xCE, 0x7E, 0x20, 0xF3, 0xBD, 0x9C, 0xDF, 0x86, 0xF4,
0x97, 0xB2, 0x55, 0xF2, 0x63, 0x89, 0xBB, 0x25, 0x7B, 0xBE, 0x38,
0x9E, 0x8D, 0xB4, 0x48, 0x4A, 0x16, 0x93, 0xBA, 0x60, 0x3C, 0xB1,
0xD1, 0x21, 0x6B, 0x32, 0x26]
    x = y = 0
```

```
    #y = x
    #x = 0
    #y = box[x]
    out = []
    for char in data:
        x = (x + 1) % 256
        y = (y + box[x]) % 256
        box[x], box[y] = box[y], box[x]
        out.append(chr((ord(char) + box[((box[x] + box[y]) %
256)])%128))

    return ('').join(out)

lst = [0x50, 0x42, 0x38, 0x4D, 0x4C, 0x54, 0x90, 0x6F, 0xFE, 0x6F,
0xBC, 0x69, 0xB9, 0x22, 0x7C, 0x16, 0x8F, 0x44, 0x38, 0x4A, 0xEF,
0x37, 0x43, 0xC0, 0xA2, 0xB6, 0x34, 0x2C]
cipher = ''

for i in lst:
    cipher += chr(i)

print cipher.encode('hex')

key = 'keykey'
print decrypt(cipher)
```



Flag：hgame{I826-2e904t-4t98-9i82}

RE: [hgame2024-week3] encrypt（动调，AES）

```
49  v34 = 83;
50  *(_DWORD *)pszAlgId = 4522049;
51  *(__m128i *)pbInput = _mm_load_si128((const __m128i *)&xmmword_7FF7C1DD34F0);
52  v36 = _mm_load_si128((const __m128i *)&xmmword_7FF7C1DD34E0);
53  if ( BCryptOpenAlgorithmProvider(&phAlgorithm, pszAlgId, 0i64, 0) >= 0
54    && BCryptGetProperty(phAlgorithm, L"ObjectLength", pbOutput, 4u, &pcbResult, 0) >= 0 )
55  {
56    v7 = *(_DWORD *)pbOutput;
57    v8 = GetProcessHeap();
58    v5 = (UCHAR *)HeapAlloc(v8, 0, v7);
59    if ( v5 )
60    {
61      if ( BCryptGetProperty(phAlgorithm, L"BlockLength", v26, 4u, &pcbResult, 0) >= 0 )
62      {
63        v9 = *(_DWORD *)v26;
64        v10 = GetProcessHeap();
65        v11 = (UCHAR *)HeapAlloc(v10, 0, v9);
66        v6 = v11;
67        if ( v11 )
68        {
69          memcpy(v11, &unk_7FF7C1DD34A0, *(unsigned int *)v26);
70          v12 = 8i64;
71          *(__m128i *)pbInput = _mm_xor_si128(
72                                  _mm_load_si128((const __m128i *)&xmmword_7FF7C1DD3500),
73                                  _mm_loadu_si128((const __m128i *)pbInput));
74          do
```

主函数挺复杂，c++的比较难看一些，应该是调用 Bcrypt 的库进行了加密，但是无法看出来是何种加密方式，于是考虑动调

```
Stack[000028D0]:0000007B9A9DF7FC db 0F7h
Stack[000028D0]:0000007B9A9DF7FD db  7Fh ;
Stack[000028D0]:0000007B9A9DF7FE db    0
Stack[000028D0]:0000007B9A9DF7FF db    0
Stack[000028D0]:0000007B9A9DF800 db  30h ; 0
Stack[000028D0]:0000007B9A9DF801 db  0Eh
Stack[000028D0]:0000007B9A9DF802 db  6Ch ; l
Stack[000028D0]:0000007B9A9DF803 db  36h ; 6
Stack[000028D0]:0000007B9A9DF804 db 0D0h
Stack[000028D0]:0000007B9A9DF805 db    1
Stack[000028D0]:0000007B9A9DF806 db    0
Stack[000028D0]:0000007B9A9DF807 db    0
Stack[000028D0]:0000007B9A9DF808 db  40h ; @
Stack[000028D0]:0000007B9A9DF809 db    0
Stack[000028D0]:0000007B9A9DF80A db    0
Stack[000028D0]:0000007B9A9DF80B db    0
Stack[000028D0]:0000007B9A9DF80C db    0
Stack[000028D0]:0000007B9A9DF80D db    0
Stack[000028D0]:0000007B9A9DF80E db    0
Stack[000028D0]:0000007B9A9DF80F db    0
Stack[000028D0]:0000007B9A9DF810 db  41h ; A
Stack[000028D0]:0000007B9A9DF811 db    0
Stack[000028D0]:0000007B9A9DF812 db  45h ; E
Stack[000028D0]:0000007B9A9DF813 db    0
Stack[000028D0]:0000007B9A9DF814 db  53h ; S
Stack[000028D0]:0000007B9A9DF815 db    0
```

在算法处动调发现为 AES，模式为 CBC

```
Stack[000028D0]:0000007B9A9DF81D db    0
Stack[000028D0]:0000007B9A9DF81E db  69h ; i
Stack[000028D0]:0000007B9A9DF81F db    0
Stack[000028D0]:0000007B9A9DF820 db  6Eh ; n
Stack[000028D0]:0000007B9A9DF821 db    0
Stack[000028D0]:0000007B9A9DF822 db  69h ; i
Stack[000028D0]:0000007B9A9DF823 db    0
Stack[000028D0]:0000007B9A9DF824 db  6Eh ; n
Stack[000028D0]:0000007B9A9DF825 db    0
Stack[000028D0]:0000007B9A9DF826 db  67h ; g
Stack[000028D0]:0000007B9A9DF827 db    0
Stack[000028D0]:0000007B9A9DF828 db  4Dh ; M
Stack[000028D0]:0000007B9A9DF829 db    0
Stack[000028D0]:0000007B9A9DF82A db  6Fh ; o
Stack[000028D0]:0000007B9A9DF82B db    0
Stack[000028D0]:0000007B9A9DF82C db  64h ; d
Stack[000028D0]:0000007B9A9DF82D db    0
Stack[000028D0]:0000007B9A9DF82E db  65h ; e
Stack[000028D0]:0000007B9A9DF82F db    0
Stack[000028D0]:0000007B9A9DF830 db  43h ; C
Stack[000028D0]:0000007B9A9DF831 db    0
Stack[000028D0]:0000007B9A9DF832 db  42h ; B
Stack[000028D0]:0000007B9A9DF833 db    0
Stack[000028D0]:0000007B9A9DF834 db  43h ; C
Stack[000028D0]:0000007B9A9DF835 db    0
Stack[000028D0]:0000007B9A9DF836 db    0
UNKNOWN 0000007B9A9DF829: Stack[000028D0]:0000007B9A9DF829 (Synchronized with RIP)
```

密文在比较处已知

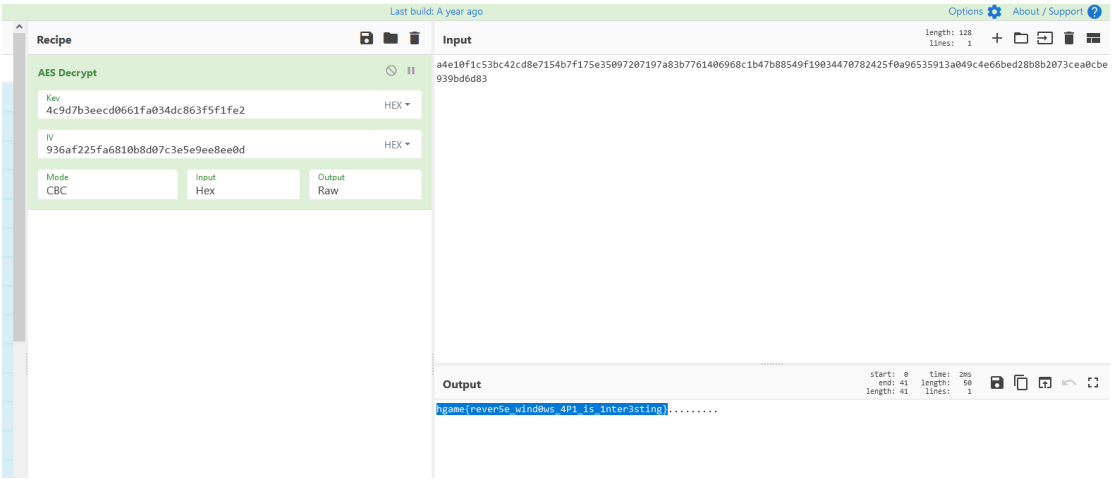
下面找 key 和 iv

```
.rdata:00007FF7C1DD34B0 ; const UCHAR pbSecret
.rdata:00007FF7C1DD34B0 pbSecret db 4Ch                 ; DATA XREF: main+1CB↑o
.rdata:00007FF7C1DD34B1 db  9Dh
.rdata:00007FF7C1DD34B2 db  7Bh ; {
.rdata:00007FF7C1DD34B3 db  3Eh ; >
.rdata:00007FF7C1DD34B4 db 0ECh
.rdata:00007FF7C1DD34B5 db 0D0h
.rdata:00007FF7C1DD34B6 db  66h ; f
.rdata:00007FF7C1DD34B7 db  1Fh
.rdata:00007FF7C1DD34B8 db 0A0h
.rdata:00007FF7C1DD34B9 db  34h ; 4
.rdata:00007FF7C1DD34BA db 0DCh
.rdata:00007FF7C1DD34BB db  86h
.rdata:00007FF7C1DD34BC db  3Fh ; ?
.rdata:00007FF7C1DD34BD db  5Fh ; _
.rdata:00007FF7C1DD34BE db  1Fh
.rdata:00007FF7C1DD34BF db 0E2h
.rdata:00007FF7C1DD34C0 db    0
.rdata:00007FF7C1DD34C1 db    0
.rdata:00007FF7C1DD34C2 db    0
.rdata:00007FF7C1DD34C3 db    0
.rdata:00007FF7C1DD34C4 db    0
.rdata:00007FF7C1DD34C5 db    0
```

```
.rdata:00007FF7C1DD349C align 20h
.rdata:00007FF7C1DD34A0 unk_7FF7C1DD34A0 db  93h        ; DATA XREF: main+15B↑o
.rdata:00007FF7C1DD34A1 db  6Ah ; j
.rdata:00007FF7C1DD34A2 db 0F2h
.rdata:00007FF7C1DD34A3 db  25h ; %
.rdata:00007FF7C1DD34A4 db 0FAh
.rdata:00007FF7C1DD34A5 db  68h ; h
.rdata:00007FF7C1DD34A6 db  10h
.rdata:00007FF7C1DD34A7 db 0B8h
.rdata:00007FF7C1DD34A8 db 0D0h
.rdata:00007FF7C1DD34A9 db  7Ch ; |
.rdata:00007FF7C1DD34AA db  3Eh ; >
.rdata:00007FF7C1DD34AB db  5Eh ; ^
.rdata:00007FF7C1DD34AC db  9Eh
.rdata:00007FF7C1DD34AD db 0E8h
.rdata:00007FF7C1DD34AE db 0EEh
.rdata:00007FF7C1DD34AF db  0Dh
.rdata:00007FF7C1DD34B0 ; const UCHAR pbSecret
.rdata:00007FF7C1DD34B0 pbSecret db 4Ch                 ; DATA XREF: main+1CB↑o
.rdata:00007FF7C1DD34B1 db  9Dh
.rdata:00007FF7C1DD34B2 db  7Bh ; {
```

iv

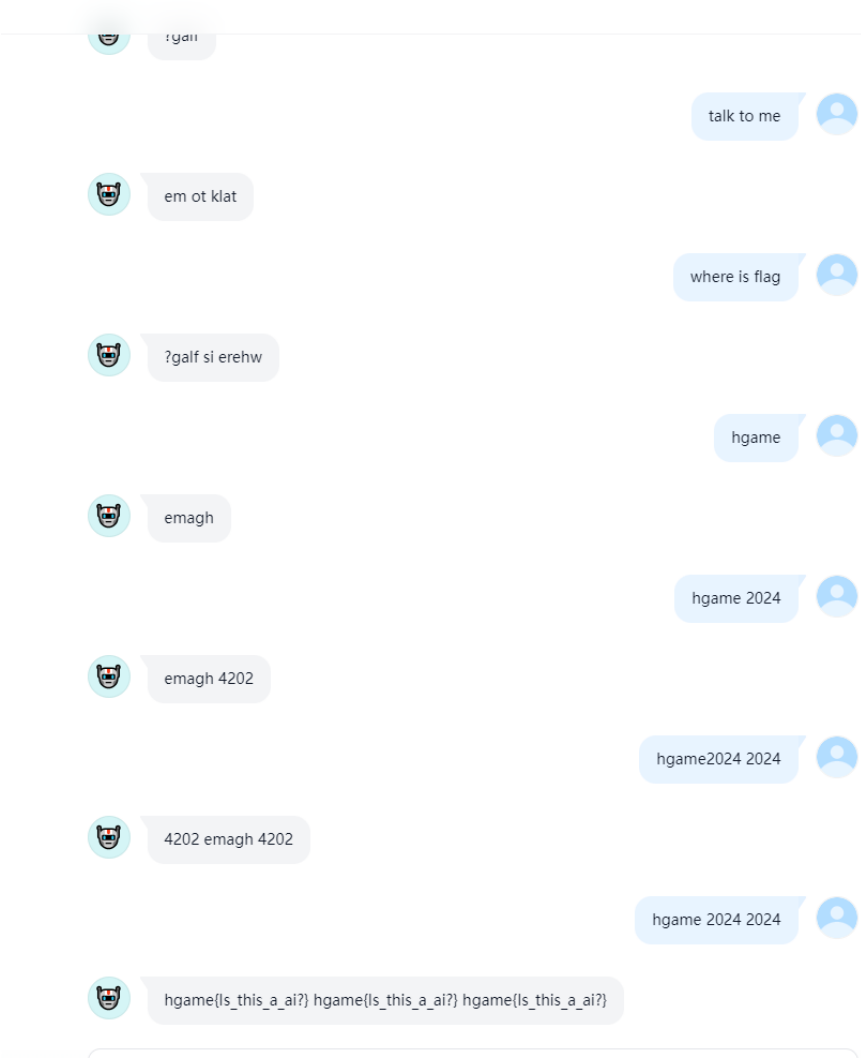
然后就丢进厨子尝试



Flag：hgame{rever5e_wind0ws_4P1_is_1nter3sting}

Misc: [hgame2024-week3] 与 ai 聊天

不知道怎么搞的，输入几次 hgame 2024，反序就不对了，然后就出 flag 了。。。



Flag：**hgame{Is_this_a_ai?}**